(2) c.1

JOINT
LOGISTICS
COMMANDERS

SAN ANTONIO I
SOFTWARE WORKSHOP

PROCEEDINGS

DECEMBER 1991

"DOD SOFTWARE FOR THE 1990s"

28 JANUARY - 1 FEBRUARY 1991

(THIS IS NOT AN APPROVED JLC DOCUMENT)

	December 1991 Proceedings
San Antonio I Software Workshop Proceedings	N/A
Joint Logistics Commanders Joint Policy Coordinating Group on Computer Resources Management, Computer Software Management Subgroup	
See Item 9	N/A
Commander Space and Naval Warfare Systems Command Code 22412 Attn: Dr Raghu Singh Washington, DC 20363-5100	
Action Plan is not available for secondary distribution	
Unlimited Distribution	

13 ABSTRACT

San Antonio I was the fifth in a series of workshops focusing on relevant software acquisition and support issues pertinent to Mission Critical Computer Resources. Workshop selectees were assigned to one of seven panels. The panels addressed the following seven issues: Software Metrics Implementation; DOD-STD-2167A and DIDs: Lessons learned/Issues; DOD-STD-2168: Lessons Learned/Issues; Computer Security/Software Integrity; Software Configuration Management; Software Reusability; and Ada secondary standards.

These Proceedings contain the final reports of each of the panels.

See the Executive Summary for summaries of the final reports of each of the panels.

14	
Software, metrics, DOD-STD-2167A, DOD-STD-2168, security, configuration management, reusability, Ada secondary standards	241

17 SECURITY CLASSIFICATION OF REPORT			
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used for announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet *optical scanning requirements*.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with ...; Trans. of ...; To be published in. ... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	-	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	-	See authorities.
NASA	-	See Handbook NHB 2200.2.
NTIS	-	Leave blank.

Block 12b. Distribution Code.

DOD	-	Leave blank.
DOE	-	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	-	Leave blank.
NTIS	-	Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

JOINT LOGISTICS COMMANDERS FIFTH SOFTWARE WORKSHOP

SAN ANTONIO I

VOLUME III

PROCEEDINGS

11 DECEMBER 1991

PRODUCED FOR THE
JOINT LOGISTICS COMMANDERS JOINT POLICY COORDINATING GROUP
ON COMPUTER RESOURCES MANAGEMENT

THIS DOCUMENT WAS PRODUCED BY THE
COMPUTER SOFTWARE MANAGEMENT SUBGROUP

FOREWORD

San Antonio I was the fifth in a series of workshops focusing on relevant software acquisition and support issues pertinent to Mission Critical Computer Resources (MCCR). The previous workshops (Monterey I, Monterey II, Orlando I, and Orlando II) were very instrumental in identifying issues that could be addressed in Department of Defense (DoD) standards for the development of mission critical systems. The central theme of San Antonio I was "DoD Software for the 1990s". Workshop selectees were assigned to one of seven panels. Each panel was assigned one particular problem area and tasked with developing solutions. The panels' conclusions reinforced the fact that more joint efforts are needed among the Services, and between DoD and industry.

The panels addressed the following seven issues:

I. Software Metrics Implementation
II. DOD-STD-2167A and DIDs: Lessons Learned/Issues
III. DOD-STD-2168: Lessons Learned/Issues
IV. Computer Security/Software Integrity
V. Software Configuration Management
VI. Software Reusability
VII. Ada Bindings

This Proceedings contains the final reports of each of these panels. The Workshop Executive Summary and Workshop Action Plan have been published as separate volumes.

Any questions concerning this material may be forwarded to:

Commander, Space and Naval Warfare Systems Command
Code 2241
Attn: Dr. Raghu Singh
Washington, DC 20363-5100

(INTENTIONAL BLANK)

TABLE OF CONTENTS

(INTENTIONAL BLANK)

1.0 INTRODUCTORY MATERIAL

1.1 INTRODUCTION

The Joint Logistics Commanders (JLC) Joint Policy Coordinating Group on Computer Resources Management (JPCG-CRM) chartered the Computer Software Management (CSM) Subgroup in January 1980. The CSM Subgroup that produced the San Antonio I Software Workshop (SA-I) consisted of:

Maj Dan Romano	Air Force Systems Command (Chair)
Dr. Raghu Singh	Space and Naval Warfare Systems Command
Douglas Gerritsen	Army Armament Research and Development Center
Donald Kosco	Air Force Logistics Command
Ralph Wootton	Marine Corps Tactical Systems Support Activity

The current CSM Subgroup members (charged with collecting, cataloging, analyzing, and reporting the recommendations from SA-I) are:

Dr. Raghu Singh	Space and Naval Warfare Systems Command (Chair)
Wayne Sherer	Army Armament Research and Development Center
Donald Kosco	Air Force Logistics Command
Capt Steve Coyne	Air Force Systems Command
Jacquelyn Nixon	Marine Corps Tactical Systems Support Activity

The JPCG-CRM organized five joint Government/industry workshops attended by computer resource professionals. Those software workshops were "Monterey I" (1979), "Monterey II" (1981), "Orlando I" (1983), "Orlando II" (1987), and "San Antonio I" (1991).

The primary purpose of SA-I was to review current software management issues common to the Services and industry, and to make specific recommendations concerning these issues. SA-I focused on acquisition management and development of MCCR. The central theme of the workshop was "DoD Software for the 1990s". SA-I identified areas offering significant cost reduction, improved system reliability, and streamlining the software acquisition and support processes.

Panel membership was limited to 12 participants, equally divided between Government and industry representatives. Panel discussions focused on the key issues within each topic area. Panel Co-Chairs were responsible for directing the discussions to the topics and questions which pertained to the accomplishment of panel objectives. Interaction between panels was encouraged to ensure that all aspects of computer software were thoroughly investigated.

Three final reports summarizing SA-I will be issued. They will be the Workshop Executive Summary, the Workshop Proceedings, and the Workshop Action Plan. The Executive Summary will capture in an optimal way what the Workshop was about and what high priority recommendations each panel made. The Proceedings will include a report from each panel detailing their recommendations. It will contain background, justification, actions required, projected benefit, estimated cost, and schedule for implementation for each recommendation. The Action Plan includes a list of prioritized workshop recommendations and defines organizational responsibilities, resources, and implementation schedules for these recommendations.

1.2 BACKGROUND

Under the sponsorship of the JLC, the JPCG-CRM hosted the SA-I Defense-Industry Software Workshop on 28 January through 1 February 1991 in San Antonio, Texas. The theme of the workshop was "DoD Software for the 1990s". The objectives were:

- to review software management and engineering issues common to the military-industrial complex,
- to learn from past experience,
- to understand future software technology, and
- to make specific recommendations for addressing these issues and technology.

The JLC is a self-chartered body endorsed by the Secretary of Defense. Members of the JLC are the commanders of:

- Army Material Command,
- Air Force Systems Command,
- Air Force Logistics Command,
- and the Deputy Chief of Naval Operations (Logistics).

The United States Marine Corps has been granted the role of an invited participant. The primary goal of the JLC is to improve military effectiveness by addressing and exploiting opportunities for joint service cooperative efforts. The JLC have been meeting regularly since 1966.

There are currently several groups under JLC supervision. Each addresses a major area of concern. One of these groups, the JPCG-CRM, specifically addresses joint DoD and industry issues and concerns related to computers--both hardware and software. Membership is comprised of representatives from the Army, the Navy, and the Air Force, with the Marine Corps sitting in as an invited participant. The JPCG-CRM has been meeting regularly since 1977.

The JPCG-CRM previously conducted four joint Defense-Industry workshops to address software related problems in the Services. These workshops were: Monterey I (1979), Monterey II (1981), Orlando I (1983), and Orlando II (1987). As a result of the Monterey I workshop, crucial needs for the consolidation of existing numerous and diverse software standards were met. The necessity for DoD-wide software standards which can be tailored was also realized. Subsequent workshops made several key recommendations to significantly improve the software standards and practices. Based on these recommendations, the JPCG-CRM published the key software standards and handbooks:

- DOD-STD-2167 (Defense System Software Development) and related Data Item Descriptions (DIDs) in June 1985;
- DOD-STD-2167A (Defense System Software Development) and related DIDs in February 1988;
- DOD-STD-2168 (Defense System Software Quality Program) and related DID in April 1988;
- MIL-HDBK-287 (Tailoring Guide for DOD-STD-2167A) in August 1989;
- MIL-HDBK-347 (MCCR Software Support) in May 1990; and
- MIL-HDBK-286 (Application Guide for DOD-STD-2168) in December 1990.

The aforementioned standards and handbooks were the first of their kind to address the software development process across DoD. They have been credited with reducing cost, improving product quality, and providing incentive to the software industry to invest in software engineering environments based on these standards. Of course, there were still unresolved issues to be addressed. Their origins were in application of the standards on real-life projects and recent technological developments. For this reason, the JPCG-CRM conducted SA-I, the fifth workshop in the series, to address these issues in the same spirit as the Monterey and Orlando workshops.

The SA-I workshop was organized into seven panels to address:

(1) Software Metrics Implementation;
(2) DOD-STD-2167A--Lessons Learned/Issues;
(3) DOD-STD-2168--Lessons Learned/Issues;
(4) Computer Security/Software Integrity;
(5) Software Configuration Management;
(6) Software Reusability; and
(7) Ada Bindings.

Each panel had both a Government Co-chair and an industry Co-chair. Panel membership, limited to 12 participants, was equally divided between Government and industry members.

The workshop generated 137 recommendations to improve software acquisition, development, engineering, and management. The text

of the panel recommendations are provided later in this document. The JPCG-CRM will review these recommendations and develop an Action Plan to implement them. The JPCG-CRM will assume several of these recommendations, particularly those related to the software standards and handbooks. For other recommendations, appropriate agencies will be requested to implement them. The Action Plan will be published separately.

2.0 PANEL FINAL REPORTS

2.1 PANEL I FINAL REPORT: SOFTWARE METRICS IMPLEMENTATION

2.1.1 INTRODUCTION

Software metrics is a structured software assessment process based upon quantitative measures of software development activities and associated software products. Software metrics, when integrated into the program management process, have proven to be effective in supporting the successful development of MCCR software systems. Implemented correctly, software metrics can be a primary tool in the management of development program cost, schedule, and technical performance.

Software metrics is a software engineering discipline and must be implemented properly within a structured framework which is consistent with the objectives of the software development program. Applied throughout all software acquisition and life cycle phases, software metrics is a quantitative decision support tool which can help the program manager objectively assess software cost, schedule, and technical risk.

To date, most MCCR software development programs have not effectively integrated software metrics assessments into required program and technical evaluations. Software metrics have been applied informally and on an irregular basis, even though there have been increasing requirements for objective management, control, and assessment of the software development processes and products. There are several key reasons for this:

- There is no DoD level policy which mandates that quantitative software measures be applied, even at a basic level, to MCCR software development programs.

- There is no consistent software metrics implementation guidance which supports the integration of software metrics technology into existing software program management and engineering disciplines.

- There is a lack of understanding, on the part of software acquisition and technical managers, of the value and limitations of software metrics technology.

To realize the considerable benefit from software metrics within the MCCR acquisition and development environment, the above issues must be effectively addressed. The purpose of this report is to provide viable recommendations to the Joint Logistics Commanders to support the near term implementation of software metrics technology.

2.1.2 OBJECTIVE

Identify, from real-life experience, a set of key software metrics which were cost effective to apply and which would provide software managers with essential information to deliver software products on time and within budget.

This objective was modified, based upon the software applications and measurement experience of the panel members, to address the key requirements for the transfer of software metrics technology into current and future MCCR software development programs. As such, the revised panel objectives focused on the generation of recommendations which addressed the issues of implementation policy and an effective and practical metrics implementation approach. It was felt that the detailed definition of specific measures would be of minimal use if they were not required to be used, or if the program and technical managers did not know how to use them.

2.1.3 BACKGROUND

2.1.3.1 SOFTWARE METRICS PROGRAM NEEDED

There are clear requirements which support the use of software metrics in MCCR software development programs. New and more complex threats demand more capable systems. These systems must be fielded on time and developed within an acquisition environment of decreasing resources. Significantly, as these systems become more functionally complex, they must meet increasingly critical quality requirements.

It is common for significant cost, schedule, and capability tradeoffs to be made late in the software development cycle of MCCR software development programs. This can be traced to the program manager being unaware of existing problems or his inability to objectively define the extent of known problems. Unfortunately, there is a decreasing tolerance for MCCR software systems which cannot meet planned cost, schedule, and technical objectives.

Recent software acquisition and engineering advances have focused on the development of higher quality software systems through the improvement of the related development, management, and engineering processes. The integration of software metrics into the development process will allow for the objective and periodic assessment of development progress, resource expenditure, and technical product quality factors. Applied software measures can be adapted to the development program characteristics, and can augment existing program and software engineering management disciplines and technologies.

Software metrics provides an objective basis for evaluating the

software development process both before the developer is selected and during actual product development. The implementation of a software metrics program will provide MCCR program managers with information to support required status assessment and decision making. Successful MCCR software developers in industry have implemented software measurement as part of an effective software development process. This should be mirrored within the DoD acquisition community.

2.1.3.2 KEY METRICS CONCEPTS

The following key concepts have been derived from the application of software metrics to actual MCCR programs. These concepts form the basis for program implementation guidance.

- Program Integration:
  Software metrics must be integrated into the program management process. Stand alone derivation and analysis of software parameter data outside of the context of program management decisions and development activities is of no use to the program and may in fact be counterproductive.

- Metrics Requirements are Issue Driven:
  The focus and prioritization of software metrics assessment efforts are driven by defined goals, issues, and decision requirements within the development program. Metrics are not collected just for the sake of collecting metrics.

- Defined, Source Level Parameter Data:
  Software metrics analysis and assessment is based on the periodic measurement of pertinent software process and product parameters which support defined development program requirements. The measurement methodology must be defined and must be consistent over time. Measurements are taken at the lowest practical process or product level. For a given parameter, the nature of the data changes over the software development life cycle. Parameter data may be expressed in the form of plans, actuals, and estimates/projections. The measurement methodology across each data type should remain consistent.

- Metrics Change During the Development Cycle:
  Software activities and products change over the development cycle. The software parameters which can be measured, therefore, also change. Early in a development much of the parameter data is available in the form of estimates and projections. As the development progresses, actual data for the same parameters becomes available. For a given parameter it

is important to compare how the projections change as well as how actuals compare to the projections.

- Metrics are Indicators, Not Absolutes:
  The quantitative results of software metrics analysis must be viewed as indicators of possible issues or conditions pertinent to the measured software development activities and products. The results cannot, given the complex interactions within the software development process, be interpreted as absolute measures of success or failure against a predefined standard. There are few, if any, deterministic measures of software progress, cost, or quality.

- Metrics Must be Understood to be Used:
  The software metrics analysis results must be conveyed to and understood by the program manager within the context of the software development characteristics and requirements. Metrics are only useful to the program manager if he/she is able to interpret them correctly with respect to the software development process and program considerations.

- Multiple Measures are Required:
  Quantitative software program assessments are based upon the use of multiple metrics. Since software development process and product attributes are related, the use of more than one software parameter or metric provides for more accurate issue definition and cross-validation of assessment findings.

2.1.4 SCOPE

The Software Implementation Panel activities were focused to generate key near-term recommendations pertinent to the implementation of software metrics technology in current MCCR programs. As such, the following factors were defined in terms of scope:

- Implementation Guidelines:
  Software metrics implementation guidelines encompass the minimum set required for successful integration of software metrics into the MCCR program management process. The guidelines support the implementation of an initial set of "core" metrics as well as the introduction of more advanced measures and analysis methods as required.

- Software Measures:
  Recommended software measures comprise a basic set of "core" measures. These measures are defined as those

required to support software development objectives and issues common to all MCCR software developments. Measures pertinent to software development progress, resource expenditure, product quality, and program stability are included. The detail of the definition of these metrics was held at a relatively high level for the purposes of the panel recommendations. This was to ensure consistency with the key concepts and associated implementation guidelines.

- Application Phase:
Although metrics are applicable throughout all software life cycle phases, the panel focused on the following periods:

  * Pre-Award Phase: This phase includes acquisition and evaluation activities which support program justification and developer selection.

  * Software Development Phase: This phase includes software development activities and products from requirements definition through system testing. These activities may occur during Demonstration/ Validation, Full Scale Development (FSD), or Post Deployment Software Support (PDSS).

These two activity/product phases are considered to be closely related with respect to software metrics implementation. Significant issues inherent during software development may be the result of constraints defined in the Pre-Award phase. During the Pre-Award phase, software metrics are used to validate the feasibility of the development program. During the Software Development Phase, software metrics are used to assess the feasibility of program changes, to quantify software development status in terms of cost, schedule, quality, and stability, and to project future expectations in these three areas.

- Software Metrics User:
The target user was defined as the Government MCCR program manager responsible for managing the cost, schedule, and technical requirements of software development. Software metrics program implementation and subsequent analysis support was assumed to be either resident in the program office or available to the program office via a Government technical agency. Developer perspectives were considered as applicable.

2.1.5 ASSUMPTIONS AND CONSTRAINTS

The recommendations of the Software Metrics Implementation Panel were developed within the context of the following assumptions and constraints:

- Software metrics programs, when applied correctly, add significant value to the management of MCCR software development programs.
- How the metrics process is implemented is more important than the individual metrics.
- Software metrics parameter data required by the Government will be specified within the development contract and be provided by the developer on a periodic basis.
- Developer impact and cost will be minimized by the selection of software developers with mature software development processes and by the implementation of metrics which are driven by program issues and requirements.
- Once the software metrics program is implemented per the recommended guidelines, the program office can adapt the software measures applied to the program based upon the software development and technical characteristics and assessment requirements.
- The core set of metrics is the minimum set required for all MCCR software development programs.
- Detailed software metrics implementation guidance, and detailed definitions of the core metrics data parameters and metrics indicator instantaneously will be developed to support program implementations should the panel recommendations be approved. (These should include a software metrics implementation guidebook and contract application requirements.)
- Program offices will be supported by Government metrics technical centers with respect to software metrics implementation and program assessment.

2.1.6 APPROACH

The challenge to the Software Metrics Implementation Panel was significant. As detailed in this report, considerable barriers exist with respect to the widespread implementation of software metrics technology within the MCCR software development community.

As software professionals with actual experience in the application of software metrics to actual programs in Government and industry, the panel members supported the modification of the panel objectives to address the current metrics issues. Again, the stated goal was to define a set of recommendations pertinent to the near-term implementation of software metrics for all MCCR software development programs.

Panel efforts initially focused on the set of implementation guidelines which needed to be conveyed to prospective program managers. There was universal commitment to the concept that software metrics must be integrated into the program management process, with metrics analysis based on low level parameter data and driven by specific program issues.

Application of specific measures can not be arbitrary, there has to be a reason for the measurement. The generation of the "core" metrics set was the second, and most difficult task. A set of common program issues were defined for both the Pre-award and Software Development phases. These issues included those in the categories of schedule progress, resource utilization, and technical product quality. Candidate "core" metrics were then derived in support of the defined issues based upon panel member experience. The "core" set was then reviewed and prioritized in panel subgroups, followed by a review by the entire panel.

The last major issue pertained to the need for a DoD level policy mandating the implementation of software metrics for MCCR software development programs. The panel was in agreement that program managers needed to be directed to implement metrics as part of their program management process. It was also expected that the continuing value of a properly implemented metrics program would be substantiated as the program matured.

A draft set of recommendations was prepared and presented to the panel for review. Various changes were considered with respect to the current metrics requirements and final drafts subsequently prepared for inclusion in the final report.

2.1.7 DETAILED REPORT

A detailed discussion of each recommendation, including implementation requirements, is provided as part of each individual recommendation in Section 2.1.8 of this report.

2.1.8 RECOMMENDATIONS

The Software Metrics Implementation Panel has defined for the JLC the following individual recommendations. These recommendations are those required to implement software metrics for MCCR software development programs. They are listed in order of priority:

- Recommendation 05-01-01:
  Establish policy at the DoD level which requires the implementation of software metrics for all MCCR software development programs.

- Recommendation 05-01-02:
  Define and establish a software metrics implementation approach applicable to MCCR software development programs which incorporates software metrics training, effective data collection and analysis procedures, and the application of a "core" metrics set for both the pre-award and software development life cycle phases.

- Recommendation 05-01-03:
  Establish a software metrics subgroup within the JPCG-CRM/CSM to develop and oversee software metrics policy for MCCR software systems.

- Recommendation 05-01-04:
  Establish a principal software metrics technical center and service dependent software metrics technical centers to serve as focal points responsible for developing MCCR software metrics technology and for supporting software metrics program implementations.

- Recommendation 05-01-05:
  Define and implement acquisition policy revisions which support the requirements of MCCR software development programs.

2.1.8.1 RECOMMENDATION 05-01-01

Establish policy at the DoD level which requires the implementation of software metrics for all MCCR software development programs.

Description:

The JLC should draft and support the implementation of acquisition directives which require that an applied software metrics program be implemented for all new DoD MCCR weapon system software development programs, to include major software upgrades. To effect this, the efforts of the Defense Acquisition Executive (DAE) should be enlisted to require the use of applied software metrics as an integral part of the program management process.

The DAE should, as a matter of policy, review key software acquisition metric parameters at major milestone reviews to validate both the feasibility and development status of all MCCR software development programs.

Substantiation:

Software metrics have proven to be effective in the successful development of MCCR software systems. Software metrics, when applied as an integrated program management tool, have provided for the objective assessment of software development cost, schedule, and technical issues.

To date there has been no consistent DoD or service policy either recommending or mandating the use of software metrics. As such, MCCR program managers have been reluctant to spend critical resources to investigate and implement software metrics programs. In effect, the value of using software metrics has not been widely realized because there is no requirement for the program managers to implement them as part of the program management process.

In many instances, software metrics have been implemented on programs already considered to be seriously at risk. Although the use of metrics on such programs is helpful, the full value of the technology is not being realized. When metrics are applied across the software development life cycle, the program manager has a continuous objective assessment tool to support feasibility assessments, tradeoff analysis, and software development cost, progress, and quality assessments. Even a basic level software metrics program can help to identify and quantify software development risk throughout the development program.

Through the review of key software metrics parameters at the major acquisition milestones, the DAE would be provided with useful insight pertinent to the feasibility and status of important software development programs. As a topic subject to DAE review, software metrics would be seriously addressed by the service acquisition organizations.

Required Actions:

Short Term Actions (up to 12-18 months)

- Establish an ad hoc JLC software metrics working group to define implementation strategy and draft prospective policy documents.

- Prepare a draft issue paper and presentation for the DAE. These items should address the key implementation issues with respect to MCCR software metrics and establish a clear need for DAE support.

Long Term Actions (past 12-18 months)

- Draft final implementation documentation for forwarding to the DAE.

- Brief Service Acquisition Executives (SAEs) and appropriate service acquisition organizations.

Impact Projection:

The successful implementation of this recommendation will result in a DoD level policy which requires that software metrics be applied to MCCR software development programs. This will modify the existing acquisition environment to provide for the effective use of software metrics as part of the software program management process.

Minimal JLC resources should be required to implement this recommendation. The key is to develop a sound implementation approach for the DAE which will be practical to establish and which will show the proven results of the use of metrics.

2.1.8.2 RECOMMENDATION 05-01-02

Define and establish a software metrics implementation approach applicable to MCCR software development programs which incorporates software metrics training, effective data collection and analysis procedures, and the application of a "core" metrics set for both the Pre-award and Software Development Life Cycle Phases.

Description:

The JLC should develop an effective software development metrics approach which will serve as the basis for implementing software metrics on MCCR software development programs. This approach should be based upon actual software metrics application experience and incorporate those procedures and measures which have proven to be effective.

The approach should be defined to support those program managers which have not previously integrated software metrics into the program management process. The approach should adaptable and expandable to address new and more complex measures and analysis techniques as required by individual program characteristics and issues.

The defined software metrics implementation approach should include the following:

- Software metrics training for service program management and related organizations.

- Software metrics applications guidance which supports the capture, analysis, evaluation, and interpretation of software parameter data.

- Software measures for application prior to contract award focused on the evaluation of prospective developer capability and program feasibility.

- Software measures for application during software development focused on the evaluation of software resources expenditures, development progress, product quality, and program stability.

These four requirements are essential to the implementation of an effective software metrics program. The development of the detailed guidance and materials to implement these requirements should be the responsibility of a technical metrics center working under the direction of the JLC.

Each of these requirements is described in detail in the following sections:

Software Metrics Training Program

The JLC should develop a software metrics training program for the Government program office and related acquisition organizations. This training program should be applications oriented and keyed towards the integration of software metrics technology into the existing program management process. Training should focus on which measures should be used and how the related parameter data should be captured, processed, and evaluated within the context of the development program.

Several key guidelines must be addressed in the training program:

- Capabilities/Limitations of Measurement:
  Measurement is an aid to decision-making. The program manager must know how software is built to know how to use and interpret metrics with respect to the software development process.

- Software Metrics Interpretation:
  Metrics are indicators, not absolutes. Reaching a conclusion usually requires more than a single number. Evaluation of the consistency and accuracy of the data, examination of multiple metrics, and integration with subjective assessments are all aspects that must be considered. A bottom-up analysis methodology provides a basis for sound metrics interpretation. This bottom-up methodology is based upon the collection and

analysis of low level software parameter data which describes development processes and products.

- Software Metrics Scope:
  Software parameter measures can be utilized to provide insight on development resources, development progress, product quality, and program stability. Software metrics are static, not dynamic, and do not provide for an assessment of system functional performance within the specified mission area.

- Cost:
  Metrics are not free, but they have net benefit. An effective software development process in itself requires the use of software measurement. As such, if a developer with a mature software development process is selected, the data required by the program office should be already available.

- Proper Use:
  Metrics should be used to assess processes and products, not people and organizations. The purpose of metrics is to understand and explain, not to blame. Punitive use of metrics leads to decreased accuracy and the unavailability of data. Development data should be treated as proprietary data.

Software Metrics Application Guidance

The JLC should develop software metrics applications guidance which provides for the capture, analysis, evaluation, and interpretation of metrics data for MCCR software development programs. For each program, the program office must ensure that a software metrics capability is in place prior to contract award. The identification of an experienced support organization to implement the program within the Government is a key component of the recommendation. The applications guidance should adhere to the following guidelines:

- Issue-driven:
  Software metrics should be designed to address specific areas of interest to the program office based upon the specific issues of the development program. Many of these issues are common across all programs (e.g., cost, schedule, resources). Therefore, a core set of metrics applicable to all MCCR software acquisition programs can be identified. The metrics program should start small, with the basic set of core metrics. The metrics program should be flexible and expandable to address additional issues which may be derived from the analysis of the core metrics set or other program inputs. These additional issues will depend upon

program size, acquisition strategy, the developer's process, cost/schedule/technical risks, and changes in these characteristics throughout the development cycle.

- Software Metrics Scope:
  Software metrics should be applied throughout the life-cycle. Since metrics are linked to a software process (development activities) or product, the parameter data supporting a particular issue changes as the life-cycle progresses. The nature of the parameter data changes from estimates to plans to actuals during the life cycle. Software metrics should be applied to multiple types of software development products such as newly developed software, Non-Developmental Item (NDI) software, Commercial Off-The-Shelf (COTS), etc.

- Software Measurement Methodology:
  Documented and consistent parameter data definitions are required within a single program. This provides a critical foundation for meaningful comparisons (e.g., plans versus actuals, between subcontractors, etc.).

- Software Parameter Data:
  Software parameter data must be carefully managed. The quality of data should be continuously reviewed to determine if it is representative of the software activity or product. Data should be accurate and consistent. Derived data is less useful than data that is a direct measure of products or a direct result of the developer's process. Parameter data should be current. Automation should be used where possible to facilitate timeliness and accuracy. On-line access to developer data that is in electronic format should be provided throughout the life-cycle.

- Software Metrics Analysis and Interpretation:
  A low-level bottom-up analysis methodology should be used to ensure traceability from software products and processes to parameter data and subsequently to metrics analysis and interpretation. Multiple software parameters and analysis techniques should be used to identify and assess issues. Subjective engineering assessments should be integrated with the quantitative metrics analysis to support ongoing issue evaluations.

- Program Integration:
  Because metrics are a program management tool, and not a comprehensive, self-contained representation of the health of a program, metrics must be integrated into the software program management process in order to be useful. Metrics assessment information must be transferred to and understood by the program manager.

Pre-Award Phase Software Measures

The Pre-award Phase of the MCCR software development program includes those acquisition and evaluation activities which support program justification and developer selection. It is during this period that critical program decisions are made which can significantly impact the degree of risk that the development program will encounter. Software metrics applied during this period support the program manager in the evaluation and reduction of risk related to developer capability and program feasibility.

A "core" metric is defined as a software parameter measure which supports the assessment of issues common to all MCCR software development programs. These are the common issues which are related to software cost, schedule, and technical quality. During the Pre-Award phase, there are two key issues which must be addressed by all program managers. These include:

- The feasibility of the software development program as planned and constrained by cost, schedule, and technical requirements.

- The relative capability of the developer to successfully build the defined software product within cost and schedule constraints.

The following "core" metrics, or metrics processes, are recommended to support assessment of these issues:

- Software Development Model
  Each program office should implement one or more software development models to assess program feasibility. These models, such as Software Life Cycle Model (SLIM), System Evaluation and Estimation of Resources (SEER), Constructive Cost Model (COCOMO), etc., should be the key tools in assessing the feasibility of the original and revised software development plans. Individual software planning parameters such as effort, development time, and software size should be evaluated through the use of the models to determine realistic cost and schedule boundaries. Planned software development productivity rates should be calculated using the models and compared to industry norms and developer performance history.

  The results of the model analysis should feed back into the program planning process and tradeoffs made until a rational and defendable development plan is defined in terms of cost, schedule, and technical requirements. Thereafter, the development models

should be used to track development progress and to evaluate subsequent planning changes.

- Software Developer Process Evaluation
  The selection of a developer which has proven to be capable of building the required software system is a key to program success. It is recommended that a software process evaluation model, such as the one developed by the Software Engineering Institute (SEI), be applied to all prospective developers. The results of this software process evaluation technique should be central to the source selection decision process, with those developers with proven software development processes given priority for award.

Another issue important to program success is the capability of the program office to properly manage the software development program. There are few measurement processes or direct measures which currently support program office assessment. It is recommended, therefore, that the JLC support the development of program office capability metrics which will aid in the evaluation of overall software program risk.

Software Development Phase Software Measures

The Software Development phase includes software development activities and products from requirements definition through system testing. These activities may occur during Demonstration/Validation, FSD, or PDSS. Software metrics applied during this phase focus on the identification of new issues and on the objective assessment of known issues throughout the development process.

There are four categories of issues pertinent to the software development phase. These include:

- Software schedule progress
- Software resource performance
- Software quality
- Software program stability

The program managers for all MCCR software developments must address the risk factors in each of these areas. The following software metrics support issue identification and assessment in the above listed areas. They are recommended as "core" metrics to be implemented by all MCCR software development programs. Due to the number of measures which support each issue, they have been separated into "primary" and "secondary" groups. The primary group is recommended for initial implementation based

upon data availability and ease of interpretation. The secondary group, although still considered to be "core" metrics, are recommended for implementation at a subsequent point in time.

- Software Schedule Progress Metrics

  * Metric: Work Unit Completion (primary). A measure of the number of software work units completed over time. A software work unit is defined as a unit of software, a software document, a software test, etc. which can be measured in terms of milestone accomplishment or completion. These may be intermediate milestones (eg, number of units designed, coded, unit tested, released to Configuration Management (CM), etc.) Plans/Actuals.

  * Metric: Software Size Allocation (primary). A measure of the change in size of the software over time as allocated to incremental development builds measured in Software Line of Code (SLOC) or number of software units. This is a measure of when the code is developed and integrated and how this allocation changes over the course of the development. Plans/Actuals.

  * Metric: Software Schedule Performance (primary). A measure of software schedule performance variances and indices over time. Measures based upon the program earned value system (DOD-INST-7000.2, "Performance Measurement for Selected Acquisitions") for software cost elements. Data derived from software work package planning, estimated values of intermediate milestones for milestones for each cost element, and actual performance.

- Software Resource Performance Metrics

  * Metric: Software Cost Performance (primary). A measure of software cost performance variances and indices over time. Measures based upon the program earned value system (DOD-INST-7000.2) for software cost elements. Data derived from software work package planning, estimated values of intermediate milestones for each cost element, and actual performance.

* Metric: Staffing Performance (primary). A measure of the number of software staff by skill type, domain experience, etc, assigned to the development program over time. Includes staff turnover measures. Plans/Actuals.

* Metric: Effort Performance (primary). A measure of the number of software staff hours applied to the program over time. Plans/Actuals.

* Metric: Software Development Productivity (secondary). A measure of software development efficiency expressed in terms of product/effort. Recommended periodic assessments based upon software development model productivity definitions. Plans/Actuals.

* Metric: Facility Utilization (secondary). A measure of the use of development facilities over time which support the development of the software. These include development workstations, computer time, test facilities, support tools, CM facilities, etc. Plans/Actuals.

- Software Quality Metrics

  * Metric: Software Defects (primary). A measure of software defects discovered and corrected over time. Includes both informal and formal defects to both documentation and code. Plans/Actuals.

  * Metric: Software Requirements Conformance (secondary). A group of software product quality metrics which may include one or more of the following, measured over time or at critical milestones.

- Requirements allocation/traceability

  * test coverage/correctness
  * standards compliance. Plans/Actuals.

- Software Program Stability Metrics

  * Metric: Requirements Size/Scope Growth (primary). A measure of the change in the number of software requirements over time. Actuals.

  * Metric: Software Size/Scope Growth (primary). A measure of the change in the scope of the software development over time expressed in SLOC or software units. Plans/Actuals.

* Metric: Project Stability (primary). A measure of the number of changes, both formal and informal, to defined software requirements, design, and product baselines over time. Actuals.

* Metric: Rework Effort (secondary). A measure of the number of hours of rework required over time to baselined software products. Actuals.

Substantiation:

The detailed requirements described above support the implementation of software metrics technology into MCCR software development programs. These requirements have been derived based upon actual experience in the application of software metrics to development programs in industry and within the DoD. This experience has shown that software measurement, when applied correctly, is effective in the identification of software development issues, the quantification of those issues, and in supporting program management decisions for corrective actions.

A key part of the detailed requirements is the application of software metrics prior to contract award. During this phase metrics are used to validate that the development plan is feasible and that the developer selected to produce the software has implemented a software development process which supports the development requirements. The objective is to use the metrics as a planning tool to define a software program which is not constrained to failure.

During software development, the "core" set of metrics will provide the program manager with the basic information required to manage the software development program. Parameters relating to software cost, schedule, quality, and stability are tracked on a periodic basis. If these measures are implemented per the defined guidelines, and the program management office is trained in their interpretation, they become important pieces of information in obtaining the status and controlling the software development effort. The metrics become particularly useful in eliminating "software ambiguities" and support the objective assessment of development issues.

Required Actions:

Short Term Actions (up to 12-18 months)

- The recommendations presented in this report should be reviewed and refined to form a basis for subsequent JLC supported metrics efforts.

- The JLC should form, even if only on a limited basis, a multi-service metrics panel to initiate the JLC metrics efforts. Panel membership should be limited to service designated representatives with software acquisition metrics experience. This group should be the foundation for the JPCG-CRM software metrics subgroup addressed in Recommendation 05-01-03.

- The JLC should identify technical support resources to initiate the development of software metrics implementation materials. These should include the following:

  * Draft policy documents
  * Implementation and training guidebooks
  * Contract requirements
  * Draft changes to DOD-STD-2167A, "Defense System Software Development"
  * Software metrics "marketing" presentations

Long Term Actions (up to 12-18 months)

- Identification and selection of MCCR software metrics pilot programs.
- Multiple program software metrics support.
- Ongoing software metrics policy/technology efforts.

Impact Projection:

The successful implementation of this recommendation will result in the development of a DoD applicable MCCR software metrics program. This program will support adaptation to meet individual service and domain requirements. Significantly, this recommendation supports the introduction of software metrics into the MCCR software acquisition activity set. The information provided by software metrics will be used by the program manager and reviewed at higher acquisition levels within DoD.

This recommendation specifically addresses how software metrics should be used and interpreted by the program manager and senior acquisition decision makers. There are current DoD initiatives which are attempting to require the use of metrics without supporting user training and education. If the user community does not see a value in the implementation of metrics, or if metrics are mandated without the proper implementation and technical support, they will never be adopted. The intent of this recommendation is to support all of the requirements necessary to make software metrics a valuable program management tool.

JLC or service identified resources will be required to develop the detailed guidance and materials inherent to this recommendation. The use of organizations currently involved in metrics technology is suggested to limit the cost impact.

2.1.8.3 RECOMMENDATION 05-01-03

Establish a software metrics subgroup within the JPCG-CRM to develop and oversee software metrics policy for MCCR software systems.

Description:

The JLC should establish a software metrics subgroup within the JPCG-CRM to oversee the integration of software metrics technology into MCCR software development programs. This subgroup, comprised of members representative of all services, should review the recommendations presented in this report and act as the focal point for their implementation. In general, the metrics subgroup should be responsible for:

- Developing recommended software metrics policy.
- Identifying resources for and overseeing the development of the materials required to support initial metrics implementations.
- Identifying possible MCCR software pilot programs for metrics introduction.
- Directing long term software metrics technology efforts.
- Educating senior service and DoD acquisition organizations with respect to software metrics.

Substantiation:

Although there are several service initiatives dealing with software metrics for MCCR programs, there is no high level DoD focal point for the development of metrics policy and the implementation of metrics technology. The establishment of a JPCG-CRM software metrics subgroup would provide for a common, high level approach to DoD software metrics implementation. The subgroup could represent the requirements of each of the services, while at the same time supporting a standardized applications approach. The subgroup would also provide an interface to industry metrics initiatives.

The subgroup would be responsible for overseeing the recommendations detailed in this report, making modifications as required. The subgroup would also be responsible for identifying

resources to support the development of metrics technology, and for directing any related metrics technical center tasking. Recommendation 05-01-04 applies.

A key objective of the JPCG-CRM software metrics subgroup should be the education of higher level DoD acquisition managers in the use of software metrics. If MCCR software program metrics initiatives are to succeed, the senior decision makers must be able to objectively interpret the data presented at high level acquisition reviews.

Required Actions:

Short Term Actions (up to 12-18 months)

- Define initial subgroup representation. This may take the form of an informal JPCG-CRM software metrics working group.

- Identify available resources and define a JLC software metrics Plan of Action and Milestones (POA&M). Develop a liaison with service and industry with respect to software metrics technology.

Long Term Actions (up to 12-18 months)

- Expand the original working group representation into a software metrics JPCG-CRM subgroup.

- Designate a software metrics technical center. Define and direct tasking.

Impact Projection:

This recommendation will result in a JLC focal point to support software metrics implementation efforts. The required actions are closely related to those identified in the other recommendations outlined in this report.

Initial efforts should require minimal JLC resources. Essentially the subgroup would support the structuring of the recommended software metrics program and the "marketing" of the program to service and DoD acquisition organizations. If the program becomes viable, the subgroup should be expanded to identify resources required to implement the software metrics program on a wider basis.

2.1.8.4 RECOMMENDATION 05-01-04

Establish a principal DoD software metrics technical center and service dependent software metrics technical centers to serve as focal points responsible for developing MCCR software metrics

technology and for supporting software metrics program implementations.

Description:

It is recommended that the JLC support the establishment of a software metrics technology center within each service. Furthermore, it is recommended that the JLC support the establishment of a principal software metrics center that is jointly supported by each of the services.

The Service software metrics centers should be established at an existing command within that Service, preferably at a site currently involved in software metrics work. The principal software metrics center should be established at an existing site that is jointly supported by the services.

The primary objective for the service software metrics technology centers is to implement software metrics and associated metrics processes within that service. Tasks should include:

- Direct technical support for software metrics to program offices.
- Integration of software metrics technology within the service acquisition environment.
- Program office software metrics training.
- Dissemination of technical information from the principal software metrics center.
- Research and development of domain specific metrics.
- Development of domain specific software metrics tools.
- Collection, processing, and analysis of long-term cross program data. Establishment of service measurement "norms".
- Definition of service specific software metrics contract and data requirements.

The primary objective for the principal DoD software metrics center is to provide a DoD focal point for software metrics and communication liaison between the services, other Government agencies, academia, and industry. Tasks should include:

- Research and development with respect to required software metrics technology.

- Dissemination of software metrics information, lessons-learned, best practices, etc.

- Software metrics technical support for each of the services.

- Integration of common software metrics technology.

- Software metrics research and development.

- Development of common software metrics tools and implementation guidelines.

- Collection, processing, and analysis of long-term cross service software parameter data.

- Technical liaison between the services and academia, industry, and other Government agencies.

Substantiation:

In order to better plan, control, and improve the DoD's acquisition of MCCR software, a long-term policy for software metrics is required. To support this, the JLC should support the establishment of software metrics technology centers for the DoD as a whole and for each service. These centers would be responsible on an ongoing basis for encouraging, implementing, and integrating software metrics into the acquisition process.

Each service has different domain specific needs and acquisition considerations. It is not realistic for a single service to support the domain specific software metrics needs of the other services. The establishment of both DoD and Service level software metrics centers would support the anticipated volume of work required as metrics are integrated into the MCCR acquisition process. In addition, the separate focus of each center would provide for specific metrics requirements to be met more efficiently.

Software metrics technology centers should be established at a DoD or Service site currently researching, developing, and applying software metrics to MCCR programs. It may not be feasible for a site or group to start a software metrics technology center from seed.

Given the above, it is recommended that a service independent site be designated and chartered as the DoD focal point for software metrics technology, and that Service metrics centers be established to support Service specific requirements. The centers should be informally organized as a group to implement DoD acquisition requirements with respect to software metrics.

Required Actions:

Short Term Actions (up to 12-18 months)

- Identify and charter a principal DoD software metrics technology center and at least two Service software metrics technology centers.

- Identify short term resources to support near term JLC software metrics objectives.

Long Term Actions (up to 12-18 months)

- Identify resources to expand software metrics center tasking in support of software metrics implementation requirements.

- Define a long term research and development plan for the metrics technology centers. Current Research & Development (R&D) requirements include the following:

  * The development of a quantitative program management office capability assessment model.

  * The development of developer software development process capability standards to support source selection decisions.

  * The development of historical software parameter standards and norms to help evaluate the feasibility and risk of projected development programs.

  * The development and validation of deterministic software acquisition measures.

Impact Projection:

Implementation of this recommendation will result in the establishment of technical focal points to support the integration of software metrics into MCCR software development programs. Given the nature of the technology, it is felt that the proposed technical centers are required if the full value of metrics is to be realized.

Initial costs could be minimized by the selection of technical centers already involved in software metrics technology. In many cases current efforts would apply directly in support of JLC software metrics objectives. R&D efforts could be deferred until initial metrics implementations have been completed.

2.1.8.5 RECOMMENDATION 05-01-05

Define and implement acquisition policy revisions which support the requirements of MCCR software development programs.

Description:

The integration of software metrics into the program management process supports the identification and assessment of software development issues. As a decision support tool, software metrics supports positive program management action to correct potential development problems and to reduce software development risk.

When applied to a software development program which has been significantly constrained by the acquisition process, software metrics becomes a technology which objectively shows that the program is likely to fail. In effect, software metrics can clearly indicate if a program has been under funded, overly specified in terms of capability, or under scheduled. In short, software metrics cannot support the successful development of a program which has had an unfeasible constrained.

In some cases metrics are seen by program management as a threat, objectively defining and projecting software cost overruns and schedule slippage. The metrics information runs counter to the objectives of "keeping the program alive" with the intention of eventually fielding the specified mission capabilities.

There is, based upon actual experience, a clear need for mitigating the cost and schedule constraints currently placed upon major MCCR software development programs. These mitigation should be implemented through high level acquisition policy revisions which focus on the approval of a justifiable software development plan in terms of software cost (resources), capability (size), schedule, and developer capability. Acquisition policy should be adapted to meet the capabilities of existing software development technology. Arbitrary program constraints should not be established.

This action would encourage the program manager to establish realistic parameters for the software development, parameters which could be tracked with the use of software metrics throughout the development process.

There are a number of specific acquisition policy revisions which are required to support successful Large-Scale Software Intensive MCCR developments. It is recommended that:

- High Level acquisition policy be revised to focus on the feasibility of the software development planning parameters for major milestone decisions.

- Metrics which support justifiable program resource, schedule, and technical requirements be applied to the Defense Systems Acquisition Review Council (DSARC)/Pre-award acquisition process.

- Rational source selections based upon quantifiable software development performance of the offerer be required.

Substantiation:

Many MCCR development failures can be directly traced to the fact that there did not exist at the time of contract award adequate program resources, schedule, and/or understanding of the technical software requirements and risks. It is felt that software metrics need to be applied during the program Pre-award phase to indicate to the Government whether or not a development is actually feasible. The results of metrics assessment during this phase should be used on all acquisition levels.

A mechanism which evaluates the following aspects of the projected development program is required:

- Software development technical feasibility.

- The program management office's capability to manage the development.

- The contractor capability to develop the software successfully.

- The acquisition contract vehicle.

This should be an addition to the DSARC process or to other current acquisition policies such as DOD-INST-5000.2,"Major System Acquisition Procedures".

A major problem in the development of MCCR systems is the capability of the contractor to implement the contract requirements. Too often, the contractor does not possess the basic experience in either the application domain or in software and systems engineering. The Government Accounting Office has documented numerous cases of this situation occurring. A primary reason for this is that there is no DoD-wide policy which emphasizes the objective evaluation of contractor MCCR software development capability.

There currently exist a number of evaluation approaches, such as the SEI Software Process Assessment approach, the AFSC Capability Review, the AFSC Grey Beard Teams, etc., that are available to evaluate the basic capability of a DoD contractor to deliver MCCR

systems. While each of these approaches has limitations, they do provide some basis for measuring contractor capability.

Attempting to apply a quantitative evaluation approach with pre-defined acceptance criteria as a matter of policy would be counterproductive. Acquisition policy for MCCR software systems should be revised, however, to reward developers which have shown to be more highly capable and those with a proven record of success. This policy would encourage the improvement of the software development capability of the industry as a whole.

Required Actions:

Short Term Actions (up to 12-18 Months)

- The SEI assessment should be encouraged to be applied by all DoD contractors involved in MCCR software developments to assess their capability to successfully deliver these types of systems. The JLC should encourage self assessments in industry. They should also support the use of the assessment results as inputs into the source selection process, as long as minimum standards were not imposed.

- The JLC should initiate via the recommended JPCG-CRM software metrics subgroup an analysis of the DSARC process to define metrics which can be used to support an assessment of technical feasibility, program office capability, contractor capability, and acquisition contract vehicle feasibility for MCCR software development programs. An overall structure to measure, compare, and trade off these requirements should be addressed.

Long Term Actions (up to 12-18 months)

- The JLC should support policy revisions which call for the establishment of software developer capability standards for MCCR software development systems.

- The JLC should revise DOD-INST-5000.2 and related DSARC acquisition policy to include a structured approach for the evaluation of overall MCCR development feasibility.

Projected Impact:

If implemented, this recommendation will result in MCCR software developments which have a greater chance of success. Key acquisition constraints will be better understood at all acquisition levels, and these will be able to be addressed early in the program life cycle.

The key issues of program feasibility and contractor capability will be objectively reviewed and understood. The DoD focus on planning and implementing successful software developments will encourage industry to improve its software development processes.

The technology and processes required to implement this recommendation are for the most part included in the other recommendations of this report. Implementation of this recommendation requires the acquisition community to revise those policies which currently constrain the successful development of MCCR software systems.

2.2 PANEL II FINAL REPORT: DOD-STD-2167A and DIDs: LESSONS LEARNED/ISSUES

2.2.1 INTRODUCTION

Revision A to DOD-STD-2167 gave the contractor freedom to select a software development method best suited to meet contractual requirements. This departure from prior software development standards was intended to encourage contractors to standardize on a software development process and also make it feasible to invest in software engineering tools and techniques. Thus software development would be less costly and the resulting software products more reliable.

The standard has been in use about three years. It has been well received by the acquisition community and contractors. Despite the shortcomings stated herein, the standard has many advantages and is certainly the best standard to use until a revision is provided. Government use of the standard has been largely limited to the procurement process; it has not yet been used extensively in the PDSS arena. However, the standard's use by contractors has resulted in its shortcomings becoming exposed. Users of the standard have made their experience known for the sake of determining how to improve the standard.

The "lessons learned" from using the standard in software development and acquisition management, were used to determine major issues to focus on for improving the standard. These issues, their basis, and recommendations for solutions to the described problems are presented in this SA-I Panel II Report. Recommendations are listed in priority order.

2.2.2 OBJECTIVE

Based on lessons learned from projects using DOD-STD-2167A and its DIDs in each of the system acquisition phases, identify problems and successes in their application and tailoring. Provide recommendations for solving the problems and capitalizing on the successes.

2.2.3 BACKGROUND

DOD-STD-2167A and its DIDs were published in February 1988. DOD-STD-2167A establishes standard terminology, a standard set of deliverables to choose from, a standard set of reviews and audits to choose from, and a standard set of software management practices that may be imposed during software development. The DIDs provide format and content requirements for the deliverables which form visible evidence of the software development, design, implementation, and verification processes. In the three years since their publication, the standard and DIDs have been tailored and implemented on a wide range of MCCR developments in all

phases of the acquisition cycle. To aid in form-fitting the standard and its DIDs to projects, a tailoring handbook, Military Handbook 287 (MIL-HDBK-287), "A Tailoring Guide For DOD-STD-2167A Defense System Software Development", was prepared and published. An update to DOD-STD-2167A is planned for 1993.

This panel was established to review the experience of the contractors and the Government in the use of DOD-STD-2167A and its DIDs. The experience is documented as "lessons learned" and coalesced into issue areas to provide a better focus on approaches to solve the problems.

The results of the panel's efforts are presented as recommendations for changes and are included in this report.

2.2.4 SCOPE

The following limiting factors were imposed to assure that issues and recommendations remained in the purview of the panel II objectives:

- Problems covered by other panels (e.g., metrics) are not addressed.
- Problems covered by tangentially related standards (e.g., safety) are not addressed.
- Problems that can be categorized as management judgement calls (e.g., schedule problems) are not addressed.
- Problems which can only be solved by training or experience (e.g., keeping design out of requirements) are not addressed.
- Problems which belong in handbooks (e.g., flow-down of requirements to subcontractors) are not addressed.

2.2.5 ASSUMPTIONS AND CONSTRAINTS

It was assumed the changes required to correct the identified problems would be included in a major revision of DOD-STD-2167A, its DIDs, and related standards. To constrain deviation from objectives, the panel focused on the system and its included software.

2.2.6 APPROACH

To determine the current software management issues and to make specific recommendations on how to improve standards, processes, and product quality, Panel II took the following approach:

- "Lessons learned" from using DOD-STD-2167A were solicited to determine problems.
- "Lessons learned" were categorized into major issues.
- Major issues were assigned to three sub-panels to work on.
- Sub-panels defined problems, made recommendations and assigned priorities to the recommendations.
- The full panel reviewed the problems and recommendations.
- A final report prepared.

2.2.7 DETAILED REPORT

2.2.7.1 SUMMARY AND DETAILED REPORT

INTRODUCTION: Due to the large number of "lessons learned" presented to this panel, they were categorized and divided among three sub-panels. The three sub-panels discussed their recommendations and put them into a priority order. Time did not allow for a full panel consensus to be reached on individual issues. Therefore, recommendations are presented in the priority order established by the sub-panels.

DOD-STD-2167A in use: The standard is widely used. There are many satisfied users. Noteworthy significant changes over DOD-STD-2167 (1985) include:

- Easier to use.
  - * An index for the standard has been included. Paragraph 5 "Detailed Requirements" has the same outline for each major subparagraph. Evaluation

Criteria Templates for software product evaluation have the same headings. Evaluation Criteria are defined in Appendix D of DOD-STD-2167A.

- More design freedom.
  - * Contractor's design standards can be defined in the Software Development Plan.
- Fewer DIDs.
  - * Nine fewer DIDs.
- Improved System/Software interface requirements.

* Paragraph 5.1 specifies the initial software/system engineering interface. Paragraph 5.8 specifies the role of software in system integration and testing.

Overview of Major Issues: Six major issues were defined for consideration by the panel. They are as follows:

- System/Software Interface
- Security
- Flexibility in the Standard
- Documentation
- Reviews
- Software Product Evaluation

System/Software Interface: The sub-panel felt that the systems interface with software needed strengthening. There were seven recommendations, including:

- Strengthen Systems Engineering support for software by establishing a JLC System Engineering Managers Group tasked to initiate the development of a new standard which formalizes the System Engineering Process and its interface with DOD-STD-2167A.
- Expand DOD-STD-2167A to include developing the software architecture and define the role of software engineering in interfacing the software with the remainder of the system.
- Include a requirement to support Systems Engineering in the preparation of a draft "System User's Manual". This will encourage early definition of user/operator

activities (especially interaction scenarios) that must be supported by software.

Security: The Security Panel presented "lessons learned using DOD-STD-2167A" to Panel II co-chairs. These considerations were then described to the full Panel II membership and the consensus was that the security community's use of DOD-STD-2167A was a major issue.

The primary issue discussed focused on "security emphasis in DOD-STD-2167A". Because security is becoming a large issue in software, the panel felt it would be beneficial to include this emphasis. It was envisioned that this could be accomplished by adding references to security requirements in the standard and in DIDs wherever it is appropriate.

Flexibility in the Standard: Flexibility in the standard is key to its viability for use on a large variety of projects. A major contribution to an understanding of the software acquisition process and an important recommendation is presented in the first recommendation which follows:

- Creation and presentation of information to manage and document the products of DoD Software Development is a process which focuses on the production and approval of formal documentation products each representing an independent slice in time relative to the development process. Thus it is critically important to evolve the software development documentation process to one which stresses the management of development information with focus directed to adequacy of acquisition agency visibility and support of development and maintenance activities. This approach needs to be synergistically coordinated with parallel initiatives such as Computer-Aided Automated Logistics Systems (CALS). Database definitions developed through this effort should be biased, not only to provide the basis for development but also to enable standardizing Computer Aided System Engineering (CASE) tool interfaces.

- A number of recommendations for changes, amplifications, and clarifications to the standard DIDs, and handbook were made for ease of use and to support a wider range of needs. In particular, it was recommended that "states", "modes", and "capabilities" be defined and their usage in the DIDs further explained.

Documentation: A large number of the "lessons learned" submitted were on documentation. Of the many recommendations made on this issue, several arose from PDSS experience and some were on changing the contents of, or adding or deleting documents from,

DOD-STD-2167A. These recommendations are discussed below, with a disclaimer that their presence in this paragraph should not be construed as emphasizing their importance over any recommendations on documentation shown in paragraph 3.2.0

- Because DID format and content requirements preclude using different terminologies and exclude classes of information, strategies which enable their inclusion should be considered - like adding annexes or appendices to the DIDs, to replace relevant sections when needed.

- PDSS activities need access to software tests and reasons for design decisions arrived at during contractor software development. Consideration should be given to the feasibility of making Software Development Files (SDFs) deliverable or provide another way to make this type of information available for PDSS.

- Consider restoring the Operational Concept Document as a document delivered to the contractor with the initial "A" Specification. These two documents should be the basis for system design and required reading for reviewers and new personnel on the project.

- Consider putting all requirements traceability in a traceability document for easy reference during PDSS.

- Consider including a DID Index which is keyed to all DID paragraph headings for easy reference whenever changes are made to any deliverable document based on a DOD-STD-2167A DID. This will make it easy to determine which other documents need to be checked for change effects.

Reviews: The recommendations for reviews were for more realistic schedules for reviews, use of incremental reviews, and consideration of ways to present information being reviewed so that it is easy to understand and not too voluminous (e.g., Critical Design Review (CDR)).

Software Product Evaluation: Most of the recommendations for Software Product Evaluation were for reviewing the definitions in Appendix D of DOD-STD-2167A and the entries in the templates (figures 4 to 10) in the standard for completeness and clarity. The most important recommendation was for the standard to stress that the Software Product Evaluation activity is continuous, not discrete - like only being performed at the completion of certain documents.

Conclusion: The panel's conclusion was that the "lessons learned" support the need for a major revision of DOD-STD-2167A, and that the revision should be more than a band-aid fix. That is, the revised standard should cover the following changes:

- Include the Software Architecture of the System.
- Evolve the software development documentation process to a condition where it adequately meets acquisition agency needs for visibility and PDSS needs for complete and easy to use software documentation - coordinate this function with the CALS initiative.

2.2.8 RECOMMENDATIONS

Due to the large number of issues addressed by this panel, recommendations were organized in categories. The categories are:

- (A) System/Software Interface
- (B) Security
- (C) Flexibility in the Standard
- (D) Documentation
- (E) Reviews
- (F) Software Product Evaluation

The recommendation number (e.g., 05-02-A01) reflects the workshop number, panel number, category letter, and the priority within the category.

SYSTEM/SOFTWARE INTERFACE

2.2.8.1 RECOMMENDATION 05-02-A01

Institute a management group under the JLC for systems engineering management. This group should initiate development of a standard which formalizes the system engineering process and its interface with DOD-STD-2167A.

Problem: The major source of software problems is inadequate systems engineering.

2.2.8.2 RECOMMENDATION 05-02-A02

For critical real time systems, add tasks to DOD-STD-2167A requiring analyses and simulation of the timing and sequencing of all system elements involved in real time processing. This should include the planned software architecture, together with its interfaces to other software and hardware, to validate the design before the entire software is coded. Both the analysis and simulation should take into account the variable (random) behavior of those interacting elements.

Problem: There currently is inadequate systems engineering tasking to help ensure that deliverable software, together with its operational environment, will successfully meet critical timelines of real time systems.

2.2.8.3 RECOMMENDATION 05-02-A03

Define tasks in DOD-STD-2167A that require measuring the capacity of the computer hardware, operating system, and any other software that will be incorporated into the system software (as defined in Air Force Regulation (AFR) 800-14, "Management of Computer Resources in Systems"), to predict bottlenecks and other potential problems that could present themselves when the deliverable software is integrated with that hardware and software. This effort should be performed as part of the process of selecting the operational environment.

Problem: In many developments, severe problems are encountered too late, when the totality of the deliverable software is integrated with its operational environment.

2.2.8.4 RECOMMENDATION 05-02-A04

Consider augmenting DOD-STD-2167A to ensure final Computer Software Configuration Item (CSCI) selection is subject to contracting agency disapproval.

Problem: Some contracting agencies do not have approval authority over contractor selection of CSCIs. This leads to the selection of CSCIs without thought to life cycle maintenance costs.

2.2.8.5 RECOMMENDATION 05-02-A05

A definition of "derived" requirements should be added to section 3.

Problem: A definition of derived requirements is needed.

2.2.8.6 RECOMMENDATION 05-02-A06

Change paragraphs 5.2.1 and 5.3.1 and figure 1 to expand the scope of Preliminary Design Reviews (PDRs) and CDRs for the environment around the CSCIs.

Work with those revising MIL-STD-1521B, "Technical Reviews and Audits for Systems, Equipments, and Computer Programs", or its replacement, to call for a system-level PDR and CDR to help ensure that detailed design has over-all system integrity.

Problem: Many problems encountered with software development are system-level problems (caused by faulty lower-level assumptions

or because something falls through a crack between software and other disciplines). Presently, MIL-STD-1521B calls for PDRs and CDRs to be conducted on an individual CSCI and Hardware Configuration Item (HWCI) basis. But the preliminary and detailed design must be reviewed by a wider body of technical specialists than just those involved in building a CSCI.

2.2.8.7 RECOMMENDATION 05-02-A07

Work with those preparing the MIL-STD-499, "Engineering Management", replacement to coordinate the system engineering tasking that is software development related to ensure proper coverage of such tasks.

Provide guidance in MIL-HDBK-287 that helps project officers avoid duplication of tasking found in DOD-STD-2167A and the systems engineering task source.

Amend section 4. Potential wording is included in attachment II-1.

Problem: Activities called for in paragraphs 5.1.2 and 5.2.2, such as allocation of requirements and preparation of the Software Requirements Specifications (SRSs) and the Interface Requirements Specification (IRS), are systems engineering tasks. There is currently underway elsewhere an effort to replace MIL-STD-499 with possibly a handbook for system engineering that could create redundant tasking in future statements of work. This "handbook only" approach should be avoided.

2.2.8.8 RECOMMENDATION 05-02-A08

Change paragraphs 5.5.2.2 and 5.6.2.1 to require verification that the source code of the Computer Software Units (CSUs) and their composition into Computer Software Components (CSCs) satisfy the design as documented in the Software Design Document (SDD).

Problem: All testing, both Formal Qualification Test (FQT) and Pre-FQT (CSU and CSC), is directed towards verifying that requirements have been met. One major step is missing (not completely because paragraph 5.5.2.2 does call for ensuring that algorithms and logic are correct). Needed is the requirement to verify that the CSU code and CSC composition satisfy the design as detailed in the SDD. This would include the list of A through K of paragraph 10.1.6.1.2.2. of the SDD DID, DI-MCCR-80012A.

SECURITY

2.2.8.9 RECOMMENDATION 05-02-B01

DOD-STD-2167A, as an "umbrella" standard, should be amended to add references to security requirements and documentation at appropriate areas in the text. The intent is to provide emphasis and to clarify the relationship of security requirements to the overall software development process.

Problem: Trusted security requirements for software are being applied to an increasing number of DoD programs. However, security requirements for software are often misunderstood, overlooked or ignored, resulting in program delays or degraded security.

FLEXIBILITY IN THE STANDARD

2.2.8.10 RECOMMENDATION 05-02-C01

Expand explanations in the handbook. Provide rationale explaining the reason for each paragraph and ramifications of its inclusion or deletion.

Problem: The standard and DIDs are not adequately tailored on procurement in spite of previous encouragement to do so.

2.2.8.11 RECOMMENDATION 05-02-C02

Carefully consider alternate methods for expressing the structure of CSCIs for the purpose of capturing design and test information with respect to alternate methodologies (e.g., functional decomposition versus object-oriented technology).

Problem: Some current design methodologies may not be compatible with the CSU/CSC structure imposed by the standard and DIDs. Benefits of requiring that CSCIs be decomposed and documented as CSCs and CSUs may be outweighed by the cost of documenting and managing software in these terms when it is contrary to design precepts.

2.2.8.12 RECOMMENDATION 05-02-C03

Develop a matrix and add to MIL-HDBK-287 to identify DIDs that would be beneficial for various types and categories of software.

Problem: There is no method of determining which DIDs are appropriate for different types or categories of software.

2.2.8.13 RECOMMENDATION 05-02-C04

The standard should be specific in terminology. Also, all four methods of qualification (inspection, analysis, demonstration, and test) should be defined unambiguously in the standard. Test should be added to the SRS DID as one of the qualification methods.

Define the following terms in the standard:

- Qualification, as noted in the terms indicated above; Qualification Level; Qualification Method; Qualification Analysis Demonstration Inspection; Qualification Test; Software Quality Factor; Qualification Test Case; Qualification Test Level; and Qualification Test Procedure.

Problem: The term "test" is used too loosely by the standard. Test is used to indicate qualification, validation, verification, and all the methods of qualification, such as inspection, analysis, demonstration, and test. There are also a number of other test related terms that are not defined.

2.2.8.14 RECOMMENDATION 05-02-C05

A better definition of CSCI is needed in DOD-STD-2167A, along with providing criteria in MIL-HDBK-287 for selection of CSCIs (In addition to the guidance in MIL-STD-483, "Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs").

Problem: No guidance is provided in DOD-STD-2167A for selecting CSCIs.

2.2.8.15 RECOMMENDATION 05-02-C06

The standard should, in paragraph 4, enjoin the contractor to comply with the government's definition of allowable non-developmental software modification, which will be stated in the Statement of Work (SOW). Also amplify in MIL-HDBK-287 that the percentage of non-developmental software that can be modified must be stated in the SOW. This percentage would be used to determine when the contractor/developer must fully document the software vs documenting only the changes.

Problem: The definition of non-developmental software does not specify how much, if any, of it may be modified and still be considered non-developmental.

2.2.8.16 RECOMMENDATION 05-02-C07

States, modes and capabilities should be defined in DIDs.

State in DIDs that if system does not have states or modes, none have to be specified.

Problem: States, modes and capabilities are not defined.

Methods of handling systems that do not have states and modes are not defined.

2.2.8.17 RECOMMENDATION 05-02-C08

Derived requirements must be defined in the standard, and must be addressed in the SRS and IRS DIDs.

Problem: Derived requirements are not addressed in the standard or DIDs.

2.2.8.18 RECOMMENDATION 05-02-C09

Retesting and "regression" testing need to be better addressed and the conditions under which they are to be performed need to be explained in the standard.

Problem: Retesting and "regression" testing are not adequately addressed in the standard.

2.2.8.19 RECOMMENDATION 05-02-C10

Add text to the foreword of the standard and reinforce in the handbook with text explaining that the life cycle model does not prescribe or preclude any software development methodologies and that alternate software development technologies are feasible within the context of the standard. Include in the handbook examples demonstrating actual mapping of various methodologies.

Problem: Inclusion of the Waterfall Model in DOD-STD-2167A is perceived as mandating a software development methodology.

2.2.8.20 RECOMMENDATION 05-02-C11

Publish a change notice incorporating MIL-HDBK-287 into "Foreword".

Problem: MIL-HDBK-287 is not referenced in DOD-STD-2167A.

2.2.8.21 RECOMMENDATION 05-02-C12

Add clarification text to the standard.

Problem: The concept of CSC and CSCI integration is ambiguous. (e.g., does CSC integration mean integration of CSUs into CSCs or CSCs into CSCIs?)

2.2.8.22 RECOMMENDATION 05-02-C13

Define "segment" as it is used in the context of DOD-STD-2167A.

Problem: The term "segment" is not defined in DOD-STD-2167A.

DOCUMENTATION

2.2.8.23 RECOMMENDATION 05-02-D01

Investigate the use of a different DID approach to solve this problem. Potential solutions are identified in attachment 2.

Problem: Detailed format and content requirements of the DIDs, coupled with DoD DID policy, are too rigid. DoD Policy only permits deleting non-applicable DID requirements. Many projects require different terminology and information that needs to be captured that is not covered by the DIDs. For example, object-oriented projects, expert systems projects, signal processing, and database management systems should use domain-specific terminology and information.

2.2.8.24 RECOMMENDATION 05-02-D02

The JPCG-CRM CSM Subgroup should form a working group to look at this problem. Potential approaches are identified in attachment II-2.

Problem: Questions have been raised about the numbers of DIDs and the placement of information within them. DIDs do not capture all important engineering information and ask for irrelevant and redundant information.

2.2.8.25 RECOMMENDATION 05-02-D03

Give guidance to contractors on possible ways of presenting information. Potential approaches are identified in attachment II-2.

Problem: Too much design detail is currently required in the SDD.

2.2.8.26 RECOMMENDATION 05-02-D04

Clarify requirements and provide concrete examples in the DIDs.

Problem: Many of the DID requirements are vague and subject to widely differing interpretations, e.g., processor characteristics in the System/Segment Design Document (SSDD), quality factor requirements in the SRS, etc.

2.2.8.27 RECOMMENDATION 05-02-D05

Remove all Requirements Traceability Matrix (RTM) fragments from existing DIDs and create a separate DID for a complete RTM that will be delivered incrementally. Include provisions for identifying and explaining the rationale for derived requirements.

Problem: There is no coherent RTM from the operational concept statement through the requirements and design documents to the test documents, showing how original and derived requirements are implemented and tested.

2.2.8.28 RECOMMENDATION 05-02-D06

Create a DID for an SDF, to include at a minimum testing information such as inputs, drivers, expected results, actual results, and test support software. Much or all of this can be on electronic media.

Problem: Information included in SDFs is being lost by the Government because the SDFs are not deliverable.

2.2.8.29 RECOMMENDATION 05-02-D07

Establish guidelines for documenting unmodified and modified existing software.

Problem: There is insufficient guidance on how existing software, unmodified or modified, should be documented when that software is used for a new system.

2.2.8.30 RECOMMENDATION 05-02-D08

Make build information, both CSCI and system, a required deliverable for all projects.

The build information must include instructions for recreating the executable software from scratch.

The Software Test Description (STD) DID needs to include government witnessing of the initialization of the system as a prerequisite for testing. Include compilation guidance and operator instruction.

Problem: Executable software formally tested and delivered to the government often cannot be recreated after delivery.

2.2.8.31 RECOMMENDATION 05-02-D09

Clarify the applicability of DOD-STD-2167A to CSCI and non-CSCI software.

Problem: It is unclear in DOD-STD-2167A, paragraph 1.2.4, when the standard applies to non-CSCI software.

2.2.8.32 RECOMMENDATION 05-02-D10

Add an appendix to DOD-STD-2167B or to MIL-HDBK-287 that is an expanded index to the DIDs. This index can include a keyword listing, as well as a list of common sections which must be consistent across the DIDs. Industry CASE tools may be under development that will do this.

Problem: Contractor personnel are not aware of the information that is required by multiple DIDs, and work is often duplicated and inconsistent as a result.

2.2.8.33 RECOMMENDATION 05-02-D11

Provide timing guidance for delivery of these documents during the development process.

Problem: There is no guidance for delivery or product evaluation of operation and support documents, the Version Description Document (VDD) or the Software Product Specification (SPS), before the end of the development process.

2.2.8.34 RECOMMENDATION 05-02-D12

Clarify figure 3 and/or the accompanying text.

Problem: There is confusion about the meaning of figure 3.

REVIEWS

2.2.8.35 RECOMMENDATION 05-02-E01

New laws passed this past year require the certification of Acquisition Management personnel. One aspect of this is a program called "ACE" training. A course should be developed and made mandatory for all personnel working with software

acquisition. This course should address the acquisition of MCCR, and other types of software, and include such topics as establishing schedules for reviews and audits for R&D programs.

Problem: A tendency of some of the services to set the dates for interim deliverables and reviews and audits frequently results in inadequate time being allocated to perform tasks, particularly system requirements analysis, systems architecture, and software requirements analysis. Products that are delivered at the culmination of these phases are often shoddy and require months to put right.

2.2.8.36 RECOMMENDATION 05-02-E02

Review the current requirements for CDR and determine the information that should really be included. In paragraph 4.1.2 encourage incremental, informal reviews. Review MIL-STD-1521B to assure adequate requirements for CDR.

Problem: The data provided prior to CDR for review, and the information presented during the CDR, overwhelms the government personnel with its volume and detail, thus the CDR becomes ineffective.

SOFTWARE PRODUCT EVALUATION

2.2.8.37 RECOMMENDATION 05-02-F01

Modify the standard to eliminate ambiguity about classifying Documentation and Design problems as Priority 1-4, as well as Priority 5. Potential solutions are identified in attachment 3.

Problem: DOD-STD-2167A assigns Priority 5 to all Documentation and Design category problems.

2.2.8.38 RECOMMENDATION 05-02-F02

Clarify and complete the definition criteria (to remove ambiguity) in appendix D of DOD-STD-2167A, and ensure definitions are provided for all criteria in figures 6-10. Potential Solutions are identified in attachment II-3.

PROBLEM: Some of the appendix D definitions of evaluation criteria are ambiguously or incompletely defined, and some of the criteria appearing in figures 6-10 are not defined in appendix D.

2.2.8.39 RECOMMENDATION 05-02-F03

Clarify the wording such that Product Evaluation is described as an on-going activity.

PROBLEM: DOD-STD-2167A implies that the Product Evaluation activities are only discrete events which immediately proceed delivery of products.

(INTENTIONAL BLANK)

ATTACHMENT II-1
SYSTEM/SOFTWARE INTERFACE
POTENTIAL SOLUTIONS

The following discussions represent potential implementations which were discussed in sub-panels. They do not represent a consensus of the full panel or the subgroups. They are provided to facilitate creative thinking in relation to Recommendation 05-02-A07.

The following represents potential verbiage for amendment of section 4:

- Add the following to 4.1 Software Development Management:

  "The contractor shall ensure software development management is consistent and coordinated with the overall system engineering management program of the contract."

- Add the following to 4.2 Software Engineering:

  "The contractor shall ensure software engineering is consistent and coordinated with the overall system engineering management program of the contract."

- Add the following to 4.3 Formal Qualification Testing:

  "The contractor shall ensure formal qualification testing is consistent and coordinated with the overall system test and evaluation program of the contract."

- Add the following to 4.4 Software Product Evaluation:

  "The contractor shall ensure software product evaluation is consistent and coordinated with the overall system quality program of the contract."

- Add the following to 4.5 Software Configuration Management:

  "The contractor shall ensure software configuration management is consistent and coordinated with the overall system configuration management program of the contract."

- Add the following to 4.6 Transition to Software Support:

  "The contractor shall ensure software support is consistent and coordinated with the overall integrated logistic support program of the contract."

ATTACHMENT II-2
DOCUMENTATION
POTENTIAL SOLUTIONS

The following discussions represent potential implementations which were discussed in sub-panels. They do not represent a consensus of the full Panel or the subgroups. They are provided to facilitate creative thinking on the recommendations they correlate to.

1. Recommendation 05-02-D01

Problem Addendum: The following are possible approaches to the problem.

There are two approaches to solving this problem that do not lead to DIDs that run fifty pages:

- Keep the static information, such as sections 1 and 2, in the DID, and provide half a dozen DOD-STD-2167A appendices for each type of technical document to choose from for different types of projects, or

- Move all DID contents to DOD-STD-2167A appendices and put the static information for each type of technical document in a top-level appendix, with changeable information in sub-appendices.

Make appropriate changes to the standard to replace CSC/CSU terminology with generic "design element" terminology, to allow for different types of projects.

Investigate the American National Standards Institute (ANSI)/Institute of Electrical and Electronics Engineers, Inc. (IEEE) Standard 1016-1987 "IEEE Recommended Practice for Software Design Descriptions," which presents a different design approach.

2. Recommendation 05-02-D02

Problem Addendum: The following are possible approaches to the problem.

Review requirements versus design information required by the DIDs. For example, the SRS DID paragraph 10.1.5.3 requires a description of CSCI internal interfaces and paragraph 10.1.5.4 asks for data element requirements. These items may impose unnecessary design restraints.

Investigate reducing the number of test documents. For example, combine the STP and STD DIDs and require a two-phased delivery so that the plan information is reviewed

at the proper time in the development process. Call the new document the Software Test Description.

Revamp the Software Test Report (STR) DID to make it a test summary only, reducing the redundancy between the STD and the STR. The STR should contain a list of the problems encountered and fielding recommendations.

Restore the DID for the Operational Concept Document (OCD) that was removed from DOD-STD-2167. Most projects find it provides extremely useful information. As it stands now, it is often difficult to find this information. In addition, being just a paragraph in the SSDD, it is not maintained.

Put all scheduling and personnel information in the SDP. Remove it from the test document DIDs. The only scheduling information that should be included in the STD is perhaps an expected duration for each test. This would make the test documents more maintainable.

Monitor the preparation of the new MIL-STD-CM. Make sure it adequately addresses software. If it does, the CM plan information currently in the SDP could be moved to the CM standard. It makes sense to have all of the CM plan in one place, and the CM standard is the logical place.

Restore the Database Design Document (DBDD) DID to highlight database information and make it more accessible. Databases are assuming a more prominent role in current systems. When this is not the case, the DBDD can be tailored out.

3. Recommendation 05-02-D03

Problem Addendum: The following are possible approaches to the problem.

Revise the SDD DID to include only high-level design information that the government requires for review and software support. Some information could perhaps be adequately covered by well commented code and SDF contents. See if some information, such as error handling, needs to be part of top-level design, or if it can be included in a new SDF DID.

Review the need for the CDR. If one is needed, review the information required for review.

ATTACHMENT II-3
SOFTWARE PRODUCT EVALUATION
POTENTIAL SOLUTIONS

The following discussions represent potential implementations which were discussed in sub-panels. They do not represent a consensus of the full Panel or the subgroups. They are provided to facilitate creative thinking on the recommendations they correlate to.

1. Recommendation 05-02-F01

The following pages describe the recommended wording changes to appendix C of DOD-STD-2167A together with the newly reworded appendix C.

Change "its documentation" to "associated documentation" because this phrase is consistent with other occurrences in DOD-STD-2167A and connotes a broader range of applicable documents.

Change "during software operation" to "in the software or associated documentation" because "software operation" connotes "software deployment" and does not address detection of documentation and design problems before the software is available for testing.

Change "supporting documentation" to "associated documentation" because "supporting documentation" connotes "software support documentation" such as that defined in paragraph 4.6.4 (Software Support and Operational Documentation) while "associated documentation" connotes a broader range of applicable documents.

Add "The documentation does not comply with or cannot be determined to comply with a) the contractual requirements, b) the baselined requirements, or c) the evaluation criteria specified in appendix D of DOD-STD-2167A." to include all sources of documentation problems.

Add "The documentation satisfies the baselined requirements but a design deficiency exists." to include design problems in the documentation.

Change "its documentation" to "associated documentation" because this phrase is consistent with other occurrences in DOD-STD-2167A and connotes a broader range of applicable documents.

Change "software problem" to "problem" so Documentation and Design category problems can be Priority 1 through 4 as well

as Priority 5 (currently they are only classified as Priority 5 which, as an example, is inappropriate for safety-related products).

Change "that does" to "that does or could do" because in complex systems (or for other reasons related to contractual as well as technical issues) a problem may be detectable only by analysis or inspection rather than demonstration and test or may be of a statistical nature.

Change "that is" to "that is or could be" because man-machine interface problems are difficult to demonstrate and quantify objectively.

PROPOSED AMENDMENT TO
APPENDIX C of DOD-STD-2167A
CATEGORY AND PRIORITY CLASSIFICATIONS FOR PROBLEM REPORTING

PURPOSE: This appendix contains requirements for a category and priority classification scheme to be applied to all problems detected in the deliverable software or associated documentation that has been placed under contractor configuration control. The requirements specified in this appendix are a mandatory part of this standard.

CLASSIFICATION BY CATEGORY: Problems detected in the software or associated documentation shall be classified by category as follows:

a. Software Problem. The software does not operate according to associated documentation and the documentation is correct.

b. Documentation Problem.

(1) The software does not operate according to associated documentation but the software is correct

(2) The documentation does not comply with or cannot be determined to comply with a) the contractual requirements, b) the baselined requirements, or c) the evaluation criteria specified in appendix D.

c. Design Problem.

(1) The software operated according to the associated documentation but a design deficiency exists.

(2) The document satisfies the baselined requirements but a design deficiency exists. The design deficiency may not

always result in a directly observable operational symptom but possesses the potential for creating further problems.

CLASSIFICATION BY PRIORITY: Problems detected in the software or associated documentation shall be classified by priority as follows:

a. Priority 1. A problem that does or could do one of the following:

(1) Prevents the accomplishment of an operational or mission essential capability specified by baselined requirements;

(2) Prevents the operator's accomplishment of an operational or mission essential capability; or

(3) Jeopardizes personnel safety.

b. Priority 2. A problem that does or could do one of the following:

(1) Adversely affects the accomplishment of an operational or mission essential capability specified by baselined requirements so as to degrade performance and for which no alternative work-around solution is known; or

(2) Adversely affects the operator's accomplishment of an operational or mission essential capability specified by baselined requirements so as to degrade performance and for which no alternative work-around solution is known.

c. Priority 3. A problem that does or could do one of the following:

(1) Adversely affects the accomplishment of an operational or mission essential capability specified by baselined requirements so as to degrade performance and for which an alternative work-around solution is known; or

(2) Adversely affects the operator's accomplishment of an operational or mission essential capability specified by baselined requirements so as to degrade performance and for which an alternative work-around solution is known.

d. Priority 4. A problem that is or could be an operator inconv nience or annoyance and which does not affect a required operational or mission essential capability.

e. Priority 5. All other errors.

Recommendation 05-02-F02

Paragraph 10.2.2. Add "and (7) the documentation is clear and unambiguous to communicate technical information to the intended audience for that document or item."

Paragraph 10.2.3. Has redundant elements with other criteria (notably 10.2.4), but doesn't really address Traceability (either upward or downward) in a technical sense. Therefore, rewrite as follows: "Traceability (as used in this standard) means the document in question is in agreement with its predecessor document(s) (i.e., nothing missing in the document in question), or that every item in the subject document has a sufficient basis in its predecessor document(s).

Satisfying the Traceability criterion in the downward sense means the subject document fully and correctly implements all the applicable stipulations of the predecessor document(s).

Satisfying the Traceability criterion in the upward sense means that every item in the subject document has a sufficient technical basis in its predecessor document(s). These items may be a direct mapping (with or without some expansion) from the predecessor document, or may be derived from a higher-level stipulation in the predecessor document, or address only a part of a higher-level stipulation. In special cases, a "computer processing or software" requirement may be validly introduced for the first time at the SRS-level in the Specification hierarchy, but these "implicit" requirements must be validated as essential, and must be specially identified/flagged in the SRS. Note: Traceability normally subsumes the "consistency with indicated documents" criterion for consistency with the predecessor document(s). See paragraph 10.2.4.

Paragraph 10.2.5 Appropriate ... techniques used. Modify last sentence to DoD: "This criterion means compliance with the techniques specified in the contract, and as described and interpreted (for implementation) in the SDP."

Paragraph 10.2.6, Appropriate allocation of ... resources. Modify as follows:

"... (1) the amount of memory of each type, processing resources, input/output resources, other computer resources, or time allocated ..."

"... (2) the sum of the allocated amounts for all subordinate elements is within the overall allocation (considering reserve capacity requirements)."

Paragraph 5.2.3. Add as last sentence: "Any SRS requirement that is not addressed by at least one test (test case) in the FQT shall be identified, flagged, and subject to Contracting Agency approval."

Paragraph 10.3. Add the undefined evaluation criteria (not in appendix D) to paragraph 10.3 (Additional Criteria):

- From figure 6

    * Adequacy of requirements allocation: CSCI to CSCs.

    * Adequacy of data recording, reduction, and analysis methods.

- From figure 7

    * Adequacy and required precision of constants.

    * Adequacy of detail in specifying test inputs. expected results, and evaluation criteria.

- From figure 8

    * Compliance with design and coding standards.

    * Compliance with maintainability requirements.

    * Compliance with CSU requirements (requirements allocated to CSU? or design?).

    * Adequacy of detail in specifying test procedures.

    * Conformance to expected results.

    * Adequacy of CSU to enter the Developmental Configuration (Note: Typo in figure 5 for IRS: Testability requirements).

- From figure 9

    * Conformance to expected results.

    * Adequacy of integrated CSCI for FQT testing.

    * Adequacy of updated test results.

- From figure 10

  * Adequacy of the tested CSCI.

    **Note: Figure 10 does not provide for evaluation of:

    a. The operation and suppo-t documents

    b. Version Description Document

    c. Software Product Specification(s)

Paragraph 4.4.4. To strengthen the use of the Definitions in appendix D, delete the word "default" in line 2 and also in line 1 of appendix D. (Sentences 2 and 3 still allow alternate definitions, subject to contracting agency approval.)

2.3 PANEL III FINAL REPORT: DOD-STD-2168: LESSONS LEARNED/ ISSUES

2.3.1 INTRODUCTION

DOD-STD-2168, "Defense System Software Quality Program", was developed as a companion document to DOD-STD-2167A. DOD-STD-2168 replaces MIL-S-52779A, "Software Quality Assurance Program Requirements". SA-I is the first time that the JLCs have reviewed the standard for documenting its benefits and shortfalls and for laying the groundwork for revision A of DOD-STD-2168.

2.3.2 OBJECTIVE

Based on lessons learned from projects using DOD-STD-2168, identify problems and successes in the application of the standard. Provide recommendations for solving problems to begin laying the framework for revision A of DOD-STD-2168.

2.3.3 BACKGROUND

DOD-STD-2168 and its Software Quality Program Plan DID were published in April 1988. DOD-STD-2168 establishes the requirements for specifying the contractor's software quality program. DOD-STD-2168 was developed with the objective of assuring the quality of (a) the deliverable software and its documentation, and (b) non-deliverable software, including non-deliverable software used in the automated manufacturing of deliverable software or in the qualification or acceptance of deliverable software. In the three years since its issue, DOD-STD-2168 and its DID have been specified on a range of MCCR software development projects.

2.3.4 SCOPE

The panel could not attribute success or failure of any specific program as a result of the use of DOD-STD-2168. Therefore, the implementation time was clearly a limiting factor in the assessment of DOD-STD-2168's impact on developmental programs. Another limiting factor in the assessment of the software quality standard is in the application of DOD-STD-2168 to those programs citing DOD-STD-7935A, "Automated Data Systems (ADS) Documentation". Although several recommendations made by the panel will improve the application of DOD-STD-2168 to Automated Information System (AIS) systems, there may be additional benefits or discrepancies that have not been addressed.

2.3.5 ASSUMPTIONS and CONSTRAINTS

Few assumptions were made during the review of the standard. Although DOD-STD-7935A is being considered for "harmonization" with DOD-STD-2167A, it was determined that DOD-STD-2168 should be

adequate for either standard. In those cases where deficiencies were found in DOD-STD-2167A, MIL-Q-9858A, "Quality Program Requirements", or other standards, the panel provided recommended changes to these standards for efficiency. Deficiencies were more readily described, as discrepancies were noted in the products received, or as practices or policies that violate common industry practices. DOD-STD-2168 has various aspects that warrant improvement.

2.3.6 APPROACH

Drawing on practical experiences with the use of DOD-STD-2168, panel members identified broad issues raised as a result of the use of DOD-STD-2168. Issues were then ranked, and restated as problems. Problems were then mapped to corresponding recommendations. Recommendations were then ranked again for subsequent prioritization.

2.3.7 DETAILED REPORT

INTRODUCTION

Quality issues were raised in seven broad areas. These areas were:

1. Metrics, Indicators and Measurement Systems
2. Role of Evaluations in Developmental Programs
3. Definition and Role of the Software Quality Program
4. Test Program Adequacy in the Quality Process
5. Interface with DOD-STD-2167A
6. Role of Documentation in the Quality Process
7. Interfaces with Other Standards

Recommendations made relating to each area are as follows:

- Incorporate a requirement for a measurement system in DOD-STD-2168 (Issue 1)
- Clarify terms, interfaces or roles (Issues 1-7)
- Improve documentation of products or processes (Issues 1-7)

2.3.8 RECOMMENDATIONS

A set of recommendations were developed in response to the issues discussed above. The recommendations were prioritized by each panel member voting for their top ten, with the top one receiving a weight of 10 and the last (of the top ten) receiving a weight of ten and the last (of the top ten) receiving a weight of one. In the final tally, the recommendations were grouped into three categories: high, medium, and lower priority.

2.3.8.1 RECOMMENDATION 05-03-01 (ISSUES: 1, 2, 3, 6)

A software quality program requires mechanisms of measurement to achieve quality. DOD-STD-2168 does not explicitly call out metrics, indicators, or the process of measurement as essential components of the software quality program. Without the use of metrics, indicators, or a process control, significant components of an effective software quality program are impossible. The terms metrics, indicators, and measurement must be defined in DOD-STD-2168, and a requirement for an effective measurement process using metrics and indicators become part of the software quality program.

Add to DOD-STD-2168 a requirement in section 4 that the software quality program includes an effective measurement process using metrics and indicators.

- mid term
- acquisition risk reducer, improves quality
- high cost for implementation of measurement program (3-5% of cost of development program)
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
  Panel VI, Reusability
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973, "Configuration Management (Draft)", and appropriate DIDs
- defined metrics in another document

Add definitions of metrics, indicators, and measurement.

- near term
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

2.3.8.2 RECOMMENDATION 05-03-02 (ISSUES: 1, 2, 3, 5)

The term "effective criteria" needs to be added to section 3.7 of DOD-STD-2168. The list of "required" evaluation criteria specified in DOD-STD-2167A, appendix D, is significantly incomplete with regard to both quality management and continuous process improvement. Nor does DOD-STD-2168 specify sufficient objective evaluation criteria for the adequacy of products or processes. Definitive evaluation criteria for the adequacy of products or processes should be included in future releases of DOD-STD-2168. This should include substantive definitions of selected criteria.

Add the requirement to evaluate adequacy of products and processes.

- mid term
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

Add a requirement in DOD-STD-2168 that the criteria for all required evaluations be defined as part of the software quality program.

- mid term
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

Add the term "effective criteria" to DOD-STD-2168, paragraph 3.7, "terms defined in DOD-STD-2167A".

- near term
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

2.3.8.3 RECOMMENDATION 05-03-03 (ISSUES: 2, 3, 6, 7)

Although paragraph 5.8 of DOD-STD-2168 requires the prime contractor to evaluate subcontractor software and documentation with respect to prime contract requirements, assure that subcontractor access is available to the prime contractor, and provide certification of subcontractor products, there are no requirements to evaluate the subcontractor capability to meet requirements prior to procurement, nor does it require a corrective action or performance rating system. The prime contractor is responsible for software developed by subcontractors. Language should be incorporated in DOD-STD-2168 that is similar to MIL-Q-9858A or MIL-STD-1535, "Supplier Quality Assurance Program Requirements", in order to assure subcontract products meet contract requirements. The standard does not adequately define these requirements at this time.

Add the following Requirements to DOD-STD-2168, paragraph 5.8, "evaluation of subcontractor management":

f. The contractor maintains a supplier/vendor rating system.

g. Contractors conduct an evaluation of the potential software subcontractors/suppliers prior to subcontract or purchase order award.

h. Contractors' corrective actions include software subcontractors/suppliers.

- mid term
- should decrease cost/risk of subcontractor effort
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

2.3.8.4 RECOMMENDATION 05-03-04 (ISSUES: 1, 2, 3, 4, 5, 6)

Software quality is an essential component of software engineering, but software engineering is not defined in DOD-STD-2167A or DOD-STD-2168. Nor is there recognition of the role of software quality in software development. DOD-STD-2167A and DOD-STD-2168 are perceived to have separate quality functions. DOD-STD-2167A requires those activities necessary to build in and design in quality. DOD-STD-2168 is interpreted to be a compliance/checking document. Overlapping and potentially conflicting requirements exist in DOD-STD-2167A and DOD-STD-2168 in the areas of evaluation records, document evaluations, corrective actions and software qualifications.

Clarification of the relationships of software quality and the development and engineering activities is needed. A corresponding clarification of the roles of these activities in the specified standards is also needed. Areas of overlap need to be resolved.

- mid term
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

2.3.8.5 RECOMMENDATION 05-03-05 (ISSUES: 2, 3, 4, 5, 6)

The present classes of qualification methods as defined by the DID for the SRS are inspection, analysis, and demonstration. The

definition of demonstration does not adequately describe the type of stimulus- response conditions to which the software should be subjected. Two types of testing methods are generally applied: GO/NO GO testing and qualification testing. Both need to be included. Change the definition of demonstration to indicate this type of testing to be GO/NO GO testing. Add a new definition which addresses quantitative testing.

Include both methods of testing in DID DI-MCCR-80025A paragraph 10.1.6.1 by changing the definition of demonstration to indicate that this is GO/NO-GO testing and add a definition for quantitative testing.

- mid term
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
- changes to DIDs of DOD-STD-2167A

2.3.8.6 RECOMMENDATION 05-03-06 (ISSUES: 1, 2, 3, 5, 6)

Section 5.3 of DOD-STD-2168 attempts to specify quality evaluation requirements for the processes of software development. However, paragraph 5.3.2, which is one sentence, is the only paragraph that addresses software development per se. All the other subparagraphs of 5.3 address activities peripheral to software development and engineering.

Expand Paragraph 5.3.2 of DOD-STD-2168 to address, and provide evaluation of, the full range of software engineering and development processes.

- mid term
- minimal costs, engineering already performed
- relates to Panel II, DOD-STD-2167A Lessons Learned

2.3.8.7 RECOMMENDATION 05-03-07 (ISSUE: 3)

Section 4.3 of DOD-STD-2168 requires the contractor to document in contractor's format the "contractor's quality program". This quality program is an overall corporate process that is not contractually required (i.e., does not apply against a single contract, but is assumed to be pre-existing). The section continues to specify that the specified quality program may be disapproved. The Government, by awarding a contract, accepts the contractor's quality program as specified in the contractor's proposal.

The corporate contractor's quality program which is assumed to be pre-existing should be used during the source selection process as one of the evaluation factors. Update MIL-HDBK-286 to reflect

the corporate quality program for source selection to recognize the potential existence of the contractor's corporate quality program.

- mid term
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

There is no need to specify in DOD-STD-2168 a requirement for the corporate program that crosses contract boundaries. DOD-STD-2168 should be reviewed to clarify the ambiguity between the corporate quality program and the contractor's quality program applied to a specific contract.

- near term
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

2.3.8.8 RECOMMENDATION 05-03-08 (ISSUES: 1, 2, 3, 5, 6, 7)

There is no definition or guidelines for the evaluation of the evaluation processes and the evaluation called out by paragraph 4.10 in DOD-STD-2168 is program unique. A section titled "Evaluation of software quality program" should be added to section 5.3.x.

Add a detailed requirement in paragraph 5 of DOD-STD-2168 to have management review of the implementation of the software quality program. This will ensure application of this requirement to each instance of the software quality program.

- mid term
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
- changes to DOD-STD-2167A and appropriate DIDs

2.3.8.9 RECOMMENDATION 05-03-09 (ISSUES: 2, 3, 4, 5)

There is no visibility into the structure (strategy, integration approach, etc) of the overall test program (formal and informal). This is especially critical for large projects invoking a number of CSCIs. Specifically missing are the incorporation of CSU testing, integration of CSUs into CSCs, integration of CSCs into CSCIs, and the integration of these efforts across numerous CSCIs in DOD-STD-2167A programs. There are no provisions for the description and planning of such tests - in either the Software

Development Plan (SDP) or the Software Test Plan (STP). The scope of the STP must be expanded to include the requirement to describe the planning for the entire software test program, to also include the test strategy. This does not imply the requirement for government approval of CSU testing.

The scope of the Software Test Plan (STP), DI-MCCR-80014, "Software Test Plan", must be expanded to include the requirement to describe the planning for the entire software test program, and to also include the test strategy. Moving informal testing requirements into the STP does not imply the requirement for government approval of informal testing (i.e., CSU testing and CSU integration testing).

- mid term
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
- changes to DOD-STD-2167A and appropriate DIDs

2.3.8.10 RECOMMENDATION 05-03-10 (ISSUES: 1, 2, 3, 4, 5, 6)

The DID for the Software Quality Program Plan (SQPP) does not require definition or description of the quality evaluation tasks to be performed. Accordingly, there is nothing in the SQPP that describes the sum total of the evaluations to be performed or the objects on which they will be performed. Section 3.3 of the SQPP needs to include a definition and description of the tasks to be performed. The DID for the software development plan in DOD-STD-2167A also needs to be expanded.

Section 3.3 of the SQPP should be revised to include a definition and description of the tasks to be performed.

- mid term
- better definition of software quality program
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

The SDP should be revised to include a definition and description of the tasks to be performed for product evaluations.

- mid term
- better definition of software quality program
- minor cost increase for documentation
- relates to Panel III, DOD-STD-2168 Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

2.3.8.11 RECOMMENDATION 05-03-11 (ISSUES: 2, 3, 4, 5)

The content of the developmental configuration, hence the product baseline, as defined in DOD-STD-2167A, includes only the Software Data Description and source code listing. The interface design information is not in the Software Product Specification. All other deliverable software engineering and test documentation is not subjected to the same level of management, quality and configuration management attention. For example, Physical Configuration Audits and Functional Configuration Audits are performed only on software data descriptions and listings, not in other important documents as well. Also, the software product specification is not adequate for PDSS. For example, the test descriptions and procedures are needed to perform regression testing after corrections and enhancements are made, and they did not receive the attention during their development that is required. This makes it difficult to structure a software quality program. The developmental configuration in DOD-STD-2167A should be expanded to include all deliverable software engineering and test documentation. This would also require redefinition of the Software Product Specification and product baseline accordingly.

The developmental configuration and product baseline need to be expanded to include all deliverable software engineering and test documentation. This is necessary in order to provide an adequate basis for PDSS.

- near term
- long term payoff
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

2.3.8.12 RECOMMENDATION 05-03-12 (ISSUES: 2, 5, 7)

The consolidation of documents in going from DOD-STD-2167 to DOD-STD-2167A (e.g., the software test procedure with the software test description) causes a configuration management problem. For example, DOD-STD-2167A requires that the Software Test Description (STD) be placed under contractor configuration control at the end of the detailed design phase. A product should be placed under configuration control only when it is reasonably stable. Since the detailed test procedures must be added into the STD at a later point in the program, the STD is not a stable product. This creates unnecessary change control and quality evaluation activity, and the possibility of introducing unauthorized changes to the previously baselined materials. The same problem occurs with the Software Design Document (SDD). In updating DOD-STD-2167 to DOD-STD-2167A there

was also a reduction in the number of DIDs for which the standard would serve as the source document. The result was that the Software Configuration Management Plan was incorporated into the SDP. Database requirements and design went into the Software Requirement Specification and SDD respectively. The Software Top-Level Design Document (STLDD), Software Detailed Design Document (SDDD), the Software Test Description (STD), and STP were combined. The effects of this streamlining are numerous. Systems with large databases are now not adequately documented. The interface design information is not in the Software Product Specification. The evolution of the software design and associated testing create an additional burden of software configuration management due to their requiring two-step documents (i.e., baselining at step 1) and the subsequent re-opening of the document to make significant changes to create the second document. Based on these lessons learned, revert to the original concept of the original version of DOD-STD-2167. Split the STD into a STD and an STP. Split the SDD into the STLDD, SDDD, and the Database Design Document (DDD).

DOD-STD-2167B should split the software design document into two software documents, a top level design and a detail design description per the original concept.

- mid term
- minor cost increase
- minimal for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

Split the STD into two separate documents, one for test descriptions and one for test procedures.

- mid term
- minor cost increase
- minimal for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

Add a DID in DOD-STD-2167A for a new document for a DDD.

- mid term
- minor cost increase
- minimal documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs

Split the software configuration management out of the SDP into a Software Configuration Management Plan.

- mid term
- minor cost increase
- minimal for documentation
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

2.3.8.13 RECOMMENDATION 05-03-13 (ISSUES: 3, 7)

DoD needs to address a training program to alert procurement personnel that MIL-S-52779A has been superseded. DOD-STD-2168 should be used for new software development contracts meeting criteria for imposing software development and quality program requirements.

A training program is required concerning embedded software acquisition. Additionally all acquisition personnel be strongly encouraged to attend.

- near term to implement
- more consistent application of software standards
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- Alternative would be a letter to all service commands detailing this position

2.3.8.14 RECOMMENDATION 05-03-14 (ISSUES: 2, 3, 6)

DI-QCIC-80572 requires a schedule be published for software quality program activities in paragraph 3.3 of the SQPP. DOD-STD-2168 does not require scheduling as part of the SQPP and thus scheduling is not a funded activity. Furthermore, there is no life cycle orientation in the SQPP. DOD-STD-2168, paragraph 4.4 should include scheduling as a part of the software quality planning process assuring a software quality program that is applicable throughout the software development life cycle.

Recommend that paragraph 4.4 of DOD-STD-2168 be revised to include a requirement for creating and updating a software quality program schedule.

- mid term
- increased continuing visibility into software quality program
- low cost increase
- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

2.3.8.15 RECOMMENDATION 05-03-15 (ISSUES: 2, 3, 4, 6, 7)

Confusion exists because of inappropriate application of MIL-Q-9858A and MIL-I-45208A, "Inspection System Requirements", to software contracts. DOD-STD-2168 interprets the requirements of MIL-Q-9858A for software development. Therefore, MIL-Q-9858A should not be applied when DOD-STD-2168 is applied to software development projects. In accordance with Federal Acquisition Regulation (FAR) requirements, MIL-I-45208A is not applicable to software development projects.

Train procurement personnel to make them aware of current policy regarding hardware and software specifications.

- near term
- increased continuing visibility into software quality program
- improved acquisition and lower cost acquisitions
- minor cost for course development

2.3.8.16 RECOMMENDATION 05-03-16 (ISSUES: 2, 5, 7)

Complex systems containing large numbers of subsystems normally result in a large number of CSCIs for DOD-2167A projects. These CSCIs normally differ substantially in size, complexity, and testing. These differences can be as large as an order of magnitude. When the software development program is prepared, it is normally designed to cover the majority of CSCIs. As a result, it becomes too generic in content for addressing complex CSCIs. As a result, quality planning is difficult. The DOD-STD-2167A SDP DID needs to be revised to address the range of complexity in software and to include a master section to cover the overall system.

The DOD-STD-2167A SDP DID should be revised to accommodate acquisitions that include multiple CSCIs of multiple complexities with description of the development strategy and approach.

- mid term
- minimal cost due to documentation

- relates to Panel II, DOD-STD-2167A Lessons Learned
  Panel V, Configuration Management
- changes to DOD-STD-2167A and appropriate DIDs
  MIL-STD-973 and appropriate DIDs

2.3.8.17 RECOMMENDATION 05-03-17 (ISSUES: 1, 2, 3, 4, 5, 6)

DOD-STD-2167A and DOD-STD-2168 are optional for Government agencies, Government development and maintenance groups are "encouraged" to use them. Their standards can be mandated only by Department of Defense Directive (DODD) or service regulation. The language was originally included to "encourage" the use and acceptance of the standards.

Service Secretaries publish a policy letter strongly encouraging the application of DOD-STD-2167A and DOD-STD-2168 on all internal software developments, including software maintenance activities.

- near term
- long term Return On Investment (ROI) resulting from consistency of software delivered to Software Support Activity (SSA)
- estimated cost is negligible
- an alternative would be to have the JLC issue a policy statement
- relates to Panel II, DOD-STD-2167A Lessons Learned

2.3.8.18 RECOMMENDATION 05-03-18 (ISSUES: 2, 3, 7)

As a result of revision, MIL-STD-1520, "Corrective Action and Disposition System for Nonconforming Material", does not apply to software. DOD-STD-2168 states that applicable requirements of MIL-STD-1520 are incorporated. This causes confusion. DOD-STD-2168 needs to be revised to incorporate changes as a result of changes to MIL-STD-1520.

Revise DOD-STD-2168 to delete reference to MIL-STD-1520.

- mid term
- minor cost increase for documentation
- relates to Panel II, DOD-STD-2168 Lessons Learned MIL-STD-1520

(INTENTIONAL BLANK)

2.4 PANEL IV FINAL REPORT: COMPUTER SECURITY/SOFTWARE INTEGRITY

2.4.1 INTRODUCTION

There is an expanding need within the DoD for high assurance that desired attributes are present in the fielded systems. The use of computers is increasing in all facets of DoD activities. Computers support financial systems, keep track of logistics, support intelligence gathering and analysis, aid command and control decision making and dissemination of orders, control communications, provide combat direction and guide weapons. Advances in technologies are dynamically applied to MCCR, increasing the complexity of the systems and the risks associated with them. DoD has become dependent on the correct operation of its computer systems to successfully perform its mission. The criticality of mission functions and the sensitivity of the data which support them are dependent upon their operational context. Criticality drives the need for high assurance.

Risks to our computer aided systems take many forms, including:

- Unauthorized disclosure of sensitive data
- Denial of service to critical functions or processors
- Contamination of critical data and processes
- Hazards to life and property

These risks are further compounded by the increasingly rich interconnection of computer systems. MCCR security must address the issues raised by networking, Data Base Management Systems (DBMS) and real-time systems technologies. The goal of computer security is to reduce these risks via the prevention of compromise, the preservation of integrity and the assurance of service.

2.4.2 OBJECTIVE

Identify basic requirements for practical methods and practices to implement computer security for MCCR systems in the various developmental, operational, and support environments.

2.4.3 BACKGROUND

To reduce risks, a high level of assurance is required that computers will function properly in the context of the mission specific system. Assurance is a life cycle concern spanning the design, development, operational, and support environments. Significant work has been done to develop techniques for assuring that sensitive data processed in computer systems is protected from unauthorized access or dissemination. DOD 5200.28-STD, DoD Trusted Computer System Evaluation Criteria (TCSEC), The Orange Book, which was issued in 1985, established levels of trust for systems that process sensitive data. These criteria are being

extended by interpretation to address networks, DBMS and real-time systems applications. The criteria include functional requirements required to support DoD Information Security Policy, as well as integrity and assurance requirements to ensure that these functions will perform correctly. These assurance requirements, rooted in rigorous requirements analysis and software engineering methods, provide a basis for establishing assurance criteria for the correct operation of computers in a wide variety of operational contexts. Specific computer functions that require a high level of assurance must be determined on a system by system basis. These functions must be derived from a thorough understanding of the computer's intended operation and can be expressed in terms of operational properties that the computer must enforce. Because of the wide application of computer systems in DoD, some requirements for high assurance systems cannot be supported by interpretations and extensions of the TCSEC. However, lessons learned in creating high assurance software can be applied.

2.4.4 SCOPE

The identification of computer security requirements is dependent on the system application. For information processing systems, a secure system is one that guarantees the integrity of and proper access to the information. For process oriented systems, such as a weapons control system, security means ensuring that the weapon is trustworthy and will perform as intended; that it is aimed correctly, that it goes where it is supposed to, that it is not inadvertently fired, and that it is resistant to in-transit countermeasures. For control systems, such as a navigation system, the security aspect may be that the system always works (reliability).

2.4.5 ASSUMPTIONS AND CONSTRAINTS

A generalized definition of computer security cannot be formulated across the range of DoD computer applications. Computer security must be defined in the context of specific missions and functions.

Computer security and software integrity are two subjects within a much larger set which requires the application of high assurance techniques across the developmental, operational, and support environments.

DOD 5200.28-STD, TCSEC, does not apply in its entirety to all systems that must be considered by this panel.

2.4.6 APPROACH

The approach adapted by the panel is described by the following steps:

- Agreed on the objectives and scope
- Reviewed the Orlando II Security Panel recommendations
- Defined and discussed the set of high level recommendations
- Broke into small groups to expand high level recommendations
- Endorsed and expanded the appropriate Orlando II recommendations
- Reviewed and modified detailed recommendations for content, consistency, coherence and clarity
- Developed section 5.2 material to augment recommendations.

2.4.7 DETAILED REPORT

Section 2.4.8 begins with the endorsement and extension of the Orlando II Security Panel recommendations. Sections 2.4.8.1 through 2.4.8.5 describe the new recommendations of this panel to the JLC, along with the associated guidelines and issues identified and discussed at the San Antonio I Software Workshop. Section 2.4.8.6 describes educational recommendations. Section 2.8.7 describes research recommendations. These sections are subdivided into separate recommendations to the JLC.

2.4.8 RECOMMENDATIONS

PANEL IV ENDORSEMENT OF ORLANDO II MCCR PANEL VIII (SECURITY) RECOMMENDATION

The Orlando II MCCR Security Panel VIII proceedings included a set of specific recommendations for the incorporation of security requirements into existing agency and military department computer security guidelines and directives. As indicated in Section 2.4.6, this panel reviewed those recommendations in order to assess progress and determine validity of any outstanding recommendations. We found that the majority of these recommendations are still valid and specific action must be taken to carry out those recommendations. The research issues proposed in the Orlando II proceedings security panel remain important goals for the PDSS environment. The recommendations identified as long term goals by the Orlando II Security Panel are now considered short term by the SA-I Computer Security / Software Integrity Panel.

Below we discuss the specific recommendations which we view as outstanding and which we endorse. Recommendations not discussed have either been fulfilled or are not directly relevant to this report.

Background

Security requirements must be defined early in the design process and be satisfied by the fielded system. This implies that designers and program managers be apprised of the need for this aspect of system design, and be directed to take the appropriate actions.

This, in turn, implies the existence of policy, which accomplishes the following:

- Highlights the need for, and high-level concern about, the problem of identifying security requirements as part of system design and analysis;
- Mandates action;
- Justifies the expenditure of effort and resources toward this end.

Policy is used to establish a need, and justify the expenditure of funds for, a particular acquisition related activity. Procedures, on the other hand, establish a uniform approach to implementing the policy. They ensure that the policy is consistently applied with a uniform level of analysis.

In the case of the identification of security requirements it is important to understand that security is as much a system requirement as performance, size and weight. Security requirements should not be identified and enumerated in a vacuum. During requirement analysis the requirements associated with security must be prioritized with aspect to all the system requirements. It is as important to avoid a secure system that does not meet its deadlines as it is to avoid a system that meets its milestones but is extremely vulnerable to threats.

Security should not be viewed as a single requirement. Security leads to many different requirements (e.g., access control, authentication). For each system, the importance of these features (both with respect to each other and to other requirements) will vary.

Scope

Computer Security Policy should apply to all missions where it is essential to maintain critical properties. The applicability of the TCSEC should be carefully defined. Policy should recognize

limitations in computer security technology while encouraging maximum use of available high assurance methods and procedures.

Assumptions and Constraints

MCCR will have a broad diversity of operational and security requirements. Although similar applications (e.g., avionics, fire control, command and control) will exhibit common requirements, each system will have a unique mix of security requirements.

Approach

The JLC should direct DOD-DIR-5200.28 (or some other appropriate high-level directive) to be amended to recognize the broad diversity of security issues, and to mandate the identification of security requirements as an integral part of the system design. In addition, DOD-STD-2167A needs to be amended to provide contractual vehicles (e.g., DIDs, Contract Data Requirements Lists (CDRLs), etc.) for specifying security attributes in system acquisitions.

Attention must be paid to other policy-level documents, notably DOD-DIR-5200.28, both for correctness, adequacy of scope, and consistency.

With regard to DOD-DIR-5200.28, this panel notes that this document is currently ambiguous with regard to scope, giving the impression of applying to all "AIS" (whereas, the TCSEC applies only to the problem of resource sharing on stand-alone multi-user computer products). Additionally, DOD-DIR-5200.28 requires that all computer systems meet an appropriate trust class in the TCSEC. The directive makes no distinctions among general purpose computers and their operating systems; mission specific systems comprised of a heterogeneous mix of computers, operating systems, application specific software, and other computer system components, such as, local area networks; application specific software running on bare hardware; or other special purpose computer systems. DoD's computer systems include all of the above in a wide variety of missions. The TCSEC trust classes are not directly applicable to many of the above computer system's descriptions. In many cases the TCSEC may be interpreted or adapted to systems to which it does not apply directly. In other cases only the assurance requirements may apply. In still others, application of the TCSEC criteria is not clear. In any case, DOD-DIR-5200.28 does not make provisions for interpreting, adapting, or not applying the TCSEC. The TCSEC provides useful insight into the development of high assurance systems, but should not be mandated across all computer architectures and mission areas. The directive, as written, results in procurement requirements which are not realizable or failure to enforce applicable portions of the TCSEC that would result in higher

assurances systems. In other words: The applicable requirements of the TCSEC are thrown out with the requirements that are not applicable. This policy stalemate, DOD-DIR-5200.28, must be reviewed and modified to clearly identify the scope of its applicability, to allow consistency between itself and DOD-STD-2167A, and to support the need for security requirements definition.

Policy must identify the role and responsibilities of the Designated Approval Authority (DAA) during the requirements definition analysis and definition phases of the system development. (See also Proceedings of Orlando II, Recommendation 4-8-03.)

Policy must address the need for risk management throughout the entire system life cycle and, in particular, during the requirements definition phase.

This panel recommends that a set of procedures be established to ensure the uniform identification of security requirements in full coordination of all other system requirements. One approach to requirements identification uses the following methodology, which has been applied to a number of mission critical systems. The methodology contains the following steps:

- Step 1, Security Concepts: Determine the security related operational concept that is to be supported or enforced by the system. This involves the identification of the sensitivity levels of the information to be processed, the clearance of users, the access control rules that are applicable and the safety and integrity goals (if any). This step identifies the initial security, safety and integrity functions that the system is to provide.

- Step 2, Preliminary Architecture: The security functions in step 1 are combined with other operational functions to determine a preliminary system architecture.

- Step 3, Risk Assessment: Conduct a security risk assessment on the preliminary architecture. This means the identification of threats facing the system, the identification and analysis of the vulnerabilities in the preliminary system design and the assessment of risk of the known threats acting on the identified vulnerabilities. In order to establish system safety and integrity requirements, a hazard analysis should be

conducted. This analysis serves to identify system states that can lead to damage or harm to property, human life, the environment or natural security.

- Step 4, Select Security Safeguards: In this step, security safeguards are selected to reduce the risks identified in step 3. Risk reduction can occur by removing or reducing vulnerabilities or by reducing the potential for loss. A trusted computer (e.g., an Evaluated Products List (EPL) product) contains a number of security safeguards that are effective in reducing risk by eliminating a number of well-known ubiquitous vulnerabilities. Moreover the selection of trusted computers is mandated by DoD Directives (DODD). However, information processing systems will have many vulnerabilities that can not be eliminated by the introduction of trusted computer products. For example, data encryption may be needed on communication links to eliminate the potential compromise of information to wiretappers. TEMPEST requirements may be required to eliminate compromising emanations.

- Step 5, Iterate Process: The steps outlined above need to be repeated many times during the system design stage. At each stage, trade-offs are conducted to determine the best way to implement each functional operational or security requirement. New designs are re-evaluated to ensure that new vulnerabilities have not been introduced or existing security safeguards made ineffectual.

The methodology outlined above is only a shell for the ultimate procedures. The methodology has to be fully incorporated with steps that identify general system requirements.

One recommended approach to providing to the Program Manager (PM) the needed security experience to execute the methodology is the establishment of a security working group at the earliest stage in the process. It is also important to identify the DAA at the beginning of the process. Ultimately, the DAA must accredit the system to process classified information. This early involvement will help to ensure that implementation trade-offs will be accreditable.

Once policy has been established, PMs need assistance in drafting the contractual vehicles to assure that security requirements are included in the procurement process. For example, the panel

recommends that the JLC distribute the DoD "Procurement Guidelines," written by Grumman Corporation, to all system acquisition offices.

Recommendation 4-8-01:
"JLC JPCG-CRM develop and coordinate a security awareness and training program for Project Managers and PDSS operational personnel."

- We endorse and further recommend that the JLC study the course "Computer Security Answers for Acquisition Managers" developed for the National Security Agency (NSA) by Booz-Allen & Hamilton, Inc., and the course "Program Manager's Guide to Computer Security" developed for NSA by Logicon, be reviewed for applicability.

- We note that there is an urgent need by project managers at all levels for increased security awareness and training program. Experience has indicated that the few initial attempts at providing such training have been very well received. Immediate steps to implement this recommendation will have high payoff in the near-term.

Recommendation 4-8-02:
"Strict systems and software engineering standards must be defined and enforced throughout the life cycle (development and post deployment) of the system."

- We endorse this recommendation and have added specific relevant guidelines and issues to be addressed in our recommendations.

Recommendation 4-8-03:
"Determine the Designated Approving Authority (DAA) at the beginning and involve the DAA throughout the life cycle of a "secure" system."

- We endorse this recommendation and discuss this further in Recommendation 05-04-02, "Certification and Accreditation".

Recommendation 4-8-04: "Risk management must be a continuous process from requirements definition throughout the life cycle."

- We endorse this recommendation and note that risk management applies to both security and programmatic concerns.

Recommendation 4-8-06: "Under the auspices of NSDD-145, the National Telecommunications and Information System and Security Committee must establish a single source for DoD computer security policy. The variety of existing DoD computer security policies and guidelines must be integrated into a single cohesive set, eliminating the confusion caused by conflicting direction."

- We endorse the basic intent of this recommendation. However, we note that NSDD-145 is no longer pertinent. Thus, we recommend examination of the steps necessary to establish DoD-wide computer security policy. A DoD-wide policy needs to be established within the context of a hierarchy of policies since the individual services will still need to interpret and implement policies specific to their mission. We further recommend that this effort also recognize efforts at the federal and international levels.

Recommendation 4-8-07:
"A mechanism for assessing the impact of a change to a system on the security of that system must be defined. A necessary part is the placement of the DAA on the configuration control board for the system."

- We endorse this recommendation and have related research recommendations documented in Recommendation 05-04-07.

Recommendation 4-8-09:
"The JLC should establish a committee to develop changes to DOD-STD-2167A that incorporate security requirements as an integral part of a systems development life cycle. The standard must include specific service requirements as well as National Computer Security Center (NCSC) requirements; it must provide DIDs to detail the required deliverables; and it should be augmented by a guidebook for application of the security standards. (A starting point for such a guideline is the "Computer Security Acquisition Management Guidebook," developed by the Space and Naval Warfare Systems Command.) The basis for this standard should proceed from an appropriate modification to DoDD's 5000.1 and 5000.2, Design to Cost/Life Cycle Cost Document."

- We strongly endorse this recommendation. We note that a number of efforts have been undertaken in this area and should be used as a basis for immediate action. Reports on these efforts are listed in the bibliography. In addition, members

of Panel IV met with members of Panel II during the San Antonio I to discuss the insertion of security policies and procedures into DOD-STD-2167A.

- As noted in the executive summary, it is imperative that program managers be provided with the contractual vehicle to ensure that security requirements be addressed through-out the life cycle of MCCR systems.

- Experience has shown that systems which must be developed under DOD-STD-2167A and which must satisfy security requirements often are not able to satisfy both sets of requirements. This leads to system acquisitions which satisfy the contractually mandated DOD-STD-2167A requirements but which fall short of satisfying critical security requirements. Ultimately such systems provide less assurance and are difficult to certify. We therefore believe that it is imperative that project managers be provided the contractual vehicle to require an integrated development approach to building secure systems under DOD-STD-2167A.

Recommendation 4-8-10:
"The services must have an organic capability to evaluate systems against trusted computing criteria and certify them for the accreditation process. (The certification process provided by the NCSC takes too long.)"

- We endorse this recommendation. However, as described in Recommendation 05-04-02, we note that there is considerable confusion surrounding the use of the terms "evaluate,""certify," and "accredit". Therefore, we suggest that the recommendation be restated as:

- "The services must have an organic capability to analyze systems against trusted computing criteria."

Recommendation 4-8-12:
"The JLC should expedite the completion and release of standard language regarding security requirements for inclusion in contracts and Statement of Work (SOW)."

- We endorse the recommendation and further discuss this in Recommendation 05-04-01.

PANEL RECOMMENDATIONS TO THE JLC

2.4.8.1 Recommendation 05-04-01

- Establish policies and procedures for defining security requirements.

- Establish policies and procedures for defining security requirements.

2.4.8.2 Recommendation 05-04-02

- Establish Certification and Accreditation policies and procedures

Background

Although "standard" definitions of the terms certification, accreditation, and evaluation are provided in the DOD-DIR-5200.28 and the TCSEC (see below), each service provides its own definitions in publications such as AFR 205-16, "Volume 1 - Computer Security Policy; Volume II - Computer Security Technical Guidance; Volume III - Computer Security General Guidance and Procedures", AFR 56-30, Army Regulation (AR) 380-19, SECNAVINST 5239.2, "Department of the Navy Automated Information Systems (AIS) Security Program", and other directives. This leads to inconsistent certification and accreditation procedures across services.

Certification and accreditation responsibilities need to be uniformly defined. Such variables as the particular type of system being developed or the type of agency doing the development influence the selection of the agency or individual responsible for certification and accreditation. The problem is compounded when a system is partially built of off-the-shelf items.

Currently, certification and accreditation responsibilities reside with the system acquisition or development agency until the program management responsibility transfers to the system's supporting agency or user. The supporting agency's life cycle responsibilities are not clearly understood and are seldom implemented properly or consistently.

Resources for maintaining centralized certification and accreditation assistance are not available. There is no central repository of certification information or skills.

Current certification and accreditation methodologies do not adequately address criticality issues, i.e., service assurance and data/system integrity. There is little available policy in this area. Standards such as DOD-DIR-5200.28 or DOD-STD-2167A do

not address criticality in systems and software development. Minimal guidance is provided for safety issues, typically nuclear safety.

Scope

This discussion expands the discussion of this issue in the Orlando II workshop. We have attempted to clarify definitions and recommend steps to achieving commonality of certification and accreditation procedures across services.

Assumptions

DOD-DIR-5200.28 delineates two types of evaluations:

- An evaluation can be performed on a computer product from a perspective that excludes the application environment; or,

- It can be done to assess whether appropriate security measures have been taken to permit the system to be used operationally in a specific environment.

The former type of evaluation is done by the NCSC through the Commercial Product Evaluation Process. The latter type of evaluation is known as a certification evaluation. It must be understood that the completion of a formal product evaluation does not constitute certification or accreditation for the system to be used in any specific application environment. On the contrary, the evaluation report only provides a trusted computer system's evaluation rating along with supporting data describing the product system's strengths and weaknesses from a computer security point of view. The system security certification and the formal accreditation procedure, done in accordance with the applicable policies of the issuing agencies, must still be followed before a system can be approved for use in processing or handling sensitive information. DAAs remain ultimately responsible for specifying security of systems they accredit.

Accreditation is defined (per DOD-DIR-5200.28) as "the official authorization that is granted to an Automatic Data Processing (ADP) system to process sensitive information in its operational environment, based upon comprehensive security evaluation of the system's hardware, firmware, and software, security design, configuration, and implementation and of the other system procedural, administrative, physical, TEMPEST, personnel, and communications security controls."

Certification is defined (per DOD-DIR-5200.28) as "the technical evaluation of a system's security features, made as part of and in support of the approval/accreditation process, that

establishes the extent to which a particular computer system's design and implementation meet a set of specified security requirements."

Approach

Because certification and accreditation are such complicated and costly endeavors, the major actions should be to standardize certification and accreditation processes across services and to define procedures to reuse certification and accreditation results when possible. The processes should include plans for acquiring the resources to implement them, education and training for personnel with certification and accreditation responsibilities, proficiency standards for those personnel, and transfer of responsibility. Implementation of the following recommendations should be performed by the Computer Security Implementation and Management Panel (CIMP) under the Joint Commanders Group on Communications-Electronics (JCG-CE).

- Develop standard certification and accreditation processes. The processes should be developed to encompass all types of systems, making allowances for small, embedded, and other unique systems. (If these are not included, system developers may decide the processes do not apply to them.)

Generic Accreditation (also known as "Type Accreditation") is a special case of accreditation where the results of one accreditation can be applied to other, similar situations. In generic accreditation, an "envelope" of performance and environmental factors is defined. Provided that the employment of a system falls within this envelope, the system can be automatically accredited. AR 380-19, SECNAVINST 5239.2, and AFR 205-16 provide for this type of accreditation, and it should be used whenever possible to minimize the cost and labor associated with accreditation.

The processes should describe when and how to accept and use other-agency evaluations and certifications, such as NCSC evaluations, Air Force Computer Security Center assessments, or certifications from other military components done under other regulations, e.g., AR 380-19. An evaluation performed in accordance with DOD 5200.28-STD should be acceptable at face value.

- Define applicability of regulations. The processes should describe the applicability of DoD and service component regulations. This is particularly critical if the system is developed under contract. If document applicability is not stated specifically in the contract, it may not be enforceable.

- Mandate process applicability. The JLC should mandate the use of the processes as applicable to all systems; this mandate should be included in the appropriate service directives. Certification and accreditation plans must define roles and responsibilities, including those in other organizations, and the inclusion of external agency evaluations. A follow-up action would be to explore the possibility of the processes becoming DoD standards.

- Develop additional guidance. Other guidance would include the applicability or inclusion of other agency (NCSC, etc.) evaluations as certification documentation or support, the acceptability of EPL reports without further testing, and responsibilities for recurrent reviews, recertification, and reaccreditation.

2.4.8.3 RECOMMENDATION 05-04-03

- Identify system design and development disciplines aimed at achieving high assurance that the system's desired attributes (e.g., security, integrity, safety) are realized in the fielded systems.

- Identify system design and implementation disciplines that yield high assurance that MCCR requirements are satisfied in fielded systems.

Background

There is a growing demand within the DoD for computer systems that enforce critical MCCR requirements, e.g., confidentiality and integrity of data and reliability of system functions. The services not only procure these systems, but they must also certify that the systems correctly enforce those requirements. The certification is successful if the developer demonstrates, to a certain degree of assurance, that the implementation enforces the critical MCCR requirements. The appropriate degree of assurance is determined by the environmental risk that the system is perceived to face.

The developer may choose among a wide variety of design techniques to construct the system; however, only a few of these techniques are suitable for constructing the demonstration, or assurance argument, that results in a successful certification. Unfortunately, the developer is not disposed to know which techniques work, because the developer does not perform the certification. Therefore, it is the responsibility of the JLC to provide, based on past experience, a list of successful techniques. While the TCSEC provides similar guidance, we have noted elsewhere that MCCR's scope and definition of security is beyond that discussed in the TCSEC. Hence, additional guidance

is necessary. This discovery is the primary motivation for the task objective stated above.

Another motivation is to identify a set of successful techniques that can be applied in the development of a large class of MCCR systems, and encourage their use. This has many benefits for the services. First, the certification effort is decreased if the certification team is familiar with the developer's techniques and has significant experience evaluating them. Second, if a technique is used frequently, there may be a demand for automated support. Automated support increases the likelihood that the technique will be applied successfully. Third, identifying and using a set of common techniques is an important step towards realizing trusted development and certification as an engineering discipline rather than a black art. Fourth, being aware of available techniques helps the service avoid ambiguity when drafting the contract.

With this guidance in hand, the developer has a greater chance of success at building a MCCR system that can be certified, and the services have a better chance of obtaining a certified system.

Scope

While the list of disciplines and techniques could include those that demonstrate hardware assurance, we are primarily concerned at this time with system design and implementation disciplines that yield software assurance.

Assumptions and Constraints

The panel believes that the techniques currently available for designing systems for evaluation against the TCSEC are also applicable for systems that must satisfy MCCR security requirements. These techniques are generally applicable to all efforts where critical requirements (not just security requirements) must be satisfied with high assurance.

An important assumption for constructing an adequate assurance argument is that requirements definition has been successfully completed (as described elsewhere in this report).

Approach

There is a growing body of research, development and analysis of high assurance trusted systems which should be used as a basis for carrying out this recommendation.

As indicated above the JLC needs to establish a catalog of techniques and approaches which can be used to provide high assurance that security requirements are satisfied in fielded systems. This is important in order to provide a "tool kit" for

system acquisition authorities and in aiding in risk identification and reduction.

It is strongly urged that as part of the implementation of this recommendation and as part of the recommendations of the education and research task (5) that information on the technical mechanisms, development approach and programmatic concerns be made available to all project managers. This information is all too often recorded only through verbal history and folklore of the companies and project offices involved in specific efforts.

It should be noted that DOD-5200.28-STD is founded on the principle of worked examples and security technology which has been "reduced to practice". Unfortunately, even within the smaller community of the NCSC and its contractors many of these "worked examples" have not been fully documented. Thus, we recommend that a joint effort between the NCSC and the JLC be undertaken to gather together descriptions of the worked examples.

In addition to the worked examples the current EPL entries should be made available to all project managers.

Finally and very importantly there are the system acquisitions which have attempted to include security requirements. Information on these efforts must be included whether they succeeded or not. Description of such efforts must include programmatic as well as technical concerns.

Steps to Implement Recommendation

Define and undertake a survey of worked examples, EPL entries, and system acquisition efforts which have employed high assurance techniques. Refer to the Products and Services Reference Catalogue.

The Questionnaire should focus on technical design and implementation techniques and programmatic concerns such as cost and schedule.

Efforts should be made to document problems and difficulties as well as successes.

The survey should be coordinated with the NCSC and the Federally Funded Research and Development Centers (FFRDCs). The role of the FFRDCs in establishing a list should not be overlooked as FFRDCs serve as independent system monitors on many system acquisitions and may be in a better position to describe the problems and difficulties than the developers or program managers.

A review committee should be established to review the initial survey questionnaire and completed responses for omissions and gross inaccuracies. We suggest that the review committee consist of prominent members of the security community as was the case with the National Research Council, "Computer at Risk" team, and current NCSC staff.

Initial List of Security Related System Acquisitions

Autodin II
Strategic Air Command Defense Information Network (SACDIN)
Korean Air Force Intelligence System (KAIS)
Tactical Reconnaissance Information Gathering System (TRIGS)
OASIS
Accat Guard
Strategic Air Command (Atlantic) SACLANT
CSSR
WMMICS Information System (WIS)
Blacker
Automated Message Processing Exchange (AMPE)
Multinet Gateway
Advanced Tactical Fighter (ATF)
Light Helicopter Experimental (LHX)

Initial List of TCSEC Worked Examples

Kernal Secure Operating System (KSOS)
Kernal Virtual Machine (KVM)
PSOS
Army Security Operating System (ASOS)
Lines of Code (K) (LOCK)
Digital Equipment Company (DEC) SKVAX

Evaluated Products

see "Products and Services Catalog."

2.4.8.4 Recommendation 05-04-04

- Identify Operational and Maintenance Policies and Procedures Ensuring the Security and Integrity of MCCR Systems.

- Establish policy and procedures for maintaining system attributes in the operational environment.

- Identify operational policies and procedures ensuring the security and integrity of MCCR systems.

- Establish policy and procedures for life cycle support aimed at maintaining desired system attributes after deployment.

2.4.8.5 Recommendation 05-04-05

- Identify life cycle support - maintenance disciplines ensuring the security and integrity of MCCR systems.

Background

System Maintenance is inevitable unless the system suffers early termination. For most MCCR maintenance is the phase that dominates the life cycle costs. It has traditionally introduced risks, particularly those associated with system degradation caused by modifications that over time diminish the integrity and clarity of the system design. Attempting to control maintenance costs and activities has been the significant driver for much of software engineering research and development. Maintenance for trusted systems is an even more challenging domain since modification to the trusted portion of the system has the potential for invalidating the certification of the system. Since implication of a modification are not readily determinable for most systems, re-evaluation and recertification necessitated by maintenance may be a significant cost and risk factor for both developers and evaluators.

Scope

The recommendations are focused on the continuance of adequate security measures during post deployment software support and operations.

Assumptions and Constraints

It is assumed that:

- Security requirements for the development phase have been correctly implemented.
- Security requirements are themselves correct.
- Hardware maintenance is done only by authorized persons.
- Physical and operational security protections have been implemented.
- Good software engineering practices are in place.

Approach

The on-going software support after the system has been deployed must be conducted with the same security measures in place as were required during development. However, the PM developing the system is not responsible for PDSS. This means that the

transition of responsibility from the developer to the support organization must occur. This transition must include the transfer of security procedures, tools and documentation. The need for this transition in turn, means that the software support organization must become involved with the developer early in tne project (i.e., when generating the SOW) to assure that the necessary planning has occurred, and that the security provisions are engineered early on.

Software support introduces new aspects to the security requirements which must be satisfied. The function of software replication and distribution, often to world-wide locations, must be properly protected. Further the introduction of the software into the user community presents new challenges that must be met. First, the user environment is much more difficult to control than the laboratory, particularly in wartime. This presents new opportunities for subversion (e.g., induction of viruses) or of unauthorized access. Second, the user will use the systems in ways unintended by the developer. This will lead to failures and requirements for software change, often under emergency conditions.

If the original design decisions included use of Non Developmental Item (NDI) that permit sharing of computer resources between special purposes applications (e.g., intelligence analysis) and general purpose applications (e.g., word processors), or permit unauthorized field modifications of special purpose applications, security can be compromised. A program of continuous evaluation and recertification must be implemented. However, the program must be carefully planned and kept to bare essentials if it is to be accepted by the system manager. The NCSC report on the Development, Operational and Maintenance Environment Security Requirements for Embedded Real Time Systems, written by the IIT Research Institute (IITRI), should be referenced for additional guidance.

In summary, the panel suggests the following:

- JLC should recommend that the program manager charters include direct responsibility for post deployment software support.

- JLC should establish a policy mandating early involvement of the software support activity and users in development programs.

- JLC should sponsor the creation of a system transition checklist, which includes security issues and analysis of operational security implications of design decisions.

- JLC should incorporate a mechanism for evaluating security provisions/deficiencies in NDI software and the NDI software development process.

2.4.8.6 Recommendation 05-04-06

- Identify and recommend specific educational needs that must be endorsed and supported.

- This section includes three individual education recommendations identified by the panel. They address different needs and should be geared towards the selected audiences and organizations.

2.4.8.6.1 Education Recommendation #1

JLC JPCG-CRM implement a training program for Program Managers.

Background

Orlando II recommendation 4-8-01 required the development and coordination of a security awareness and training program for project managers and PDSS operational support. A course was developed title "Computer Security Answers for Acquisition Managers" for National Security Agency (NSA) (by Booz Allen and Hamilton Inc.). Additionally, Logicon, Inc. developed a "Program Managers Guide to Computer Security" for NSA. These two efforts have not produced a continuing development program in this field.

Scope

This training program should focus on the development of program management personnel in the procedures needed to procure a trusted computer system.

Assumptions and Constraints

This program should build on the work already initiated in this area. See the list of documents in the Panel IV report.

Approach

This program of training should be set up and available to all DoD components. It should be supported through an institution, such as, Department of Defense Computer Institute (DODCI) or other principal training elements in DoD. The course(s) should be held regularly and should be updated as technology evolves.

2.4.8.6.2 Education Recommendation #2

Establish a security requirement early for technical security education throughout the life cycle of MCCR system developments.

Background

Technical security education and integration of security engineering into the system life cycle process are not traditionally recognized as important aspects of developing secure systems. There are no requirements to educate management and technical personnel on the impacts of security to the system development. Security policy and requirements must be clearly communicated in terms of management support, technical activities and trade-off approaches for accomplishing a secure system that satisfies mission needs. Poor communication, lack of understanding, isolation of security engineering teams, de-emphasis of security and ignorance of the role of security in the overall system can lead to serious failure, accreditation non-compliance or excessive costs. Early security education is an essential ingredient to ensure the successful development, operation and maintenance of secure MCCR systems.

Scope

The security education requirement applies to all project personnel including the Government, Government support contractors and the development contractors. The requirement for a technical security education within a secure MCCR project needs to be DoD-wide. All projects requiring trust security certification and system accreditation have security policy and requirements that must be made compatible with the overall mission requirements.

Assumptions and Constraints

Individual projects do implement a physical/procedural security education program, and some implement a technical security education program for the system engineers and developers. However, there is no mandate to educate personnel, and there is seldom any program to educate project management and other government personnel. Early education will facilitate the communication of security issues throughout the life cycle, and support engineering trade off decisions.

Approach

Define the technical security education requirement as part of the overall security engineering program. To support the integration of security engineering activities and products in a trusted MCCR development life cycle, incorporate technical

security education in the appropriate DoD standard or guidance/policy document to ensure DoD-wide implementation.

2.4.8.6.3 Education Recommendation #3

Develop an awareness of computer security technology, ethics, and Information Security (INFOSEC) requirements in universities, military academies, and other education activities.

Background

Security awareness cannot start when someone has the job to develop a secure system. The principles of security, trust, and ethics must be taught to the user communities early in their development. The need to educate users in the technology and its application cannot start too early.

Scope

Curriculum and courses that stress computer security should become a standard part of curriculum at universities, military academies, and other schools that develop the DoD community.

Assumptions and Constraints

Lack of understanding of the requirements for security, the approaches to security and the policies are deterrents to developing and using security technology.

Approach

Universities, military academies, and other educational institutions should develop courses that address computer security requirements, risk analysis, trusted systems, and ethics. This should be undertaken the same way that Ada will be developed in educational curriculum.

2.4.8.7 Recommendation 05-04-07

- Identify and recommend specific research needs that must be endorsed and supported.

Similar to the previous sections, which subdivided the education recommendation into a number of individual educational needs, this section is divided into separate areas in security that require additional research and are crucial to the advancement of security and necessary for the implementation of security requirements for advanced MCCR applications.

2.4.8.7.1 Research Recommendation #1

- Identify and investigate security requirements relevant to the use of the Ada programming language for the implementation of MCCR systems.

Background

Security requirements are mandated as described in DOD-DIR-5200.28. Likewise, the use of Ada in defense projects has been mandated by the DoD for MCCR projects. As the language matures and is extended to meet the needs of the industry and government, the ability to justify a waiver to use a different programming language (or the need to) has diminished; making a waiver more of the exception rather than the rule.

The Ada language is currently in the American National Standards Institute (ANSI) review process. The Ada 9X Project Office collected over 800 Revision Requests (RRs) in 1989. These RRs were analyzed and used to derive the Revision Issues (RIs) given to the Requirements Team (the Software Engineering Institute) to define the Ada 9X requirements. The Mapping Team (Intermetrics) is currently working on the design. In addition to the various organized Ada 9X review organizations (Distinguished Reviewers, Technical Advisory Group, 9X Canvasser Group, etc.), Ada 9X has held public reviews at various workshops and conferences at each phase of the project. Now is the time to influence the language to address the critical security requirements of MCCR applications! Some of the questions that need to be answered include:

- How does Ada's Run-time System (RTS) relate to a secure Operating System (OS)? The Ada language encapsulates many features (e.g., tasking, memory management, etc.) that are normally considered OS functions. Should the Ada Run-time be a part of the Trusted Computer Base (TCB), or should the TCB be a part of the Ada Run-time? Should they instead be separate? Can a TCB be written in Ada?

- How does the use of Ada aid or hinder the implementation of security requirements for MCCR applications? Language features like strong typing could conceivably enable security (e.g., access control mechanisms) and aid in the certification of the system. However, other features, like the abort statement, can cause severe security problems and impede security.

- How can the Ada language revision help support security? The Ada 9X Project has identified several "annexes" to include application specific requirements that will be treated similar to secondary standards;

these include: Real-Time, Distributed, Parallel Processing, Object Oriented, and Security requirements. Very limited attention has been given to the Security Annex at this time. It is essential for MCCR applications that plan to adhere to the Ada mandate to get involved and ensure that Ada 9X addresses the security and other requirements identified in this workshop.

The SIGAda Ada Run-time Environment Working Group (ARTEWG) Security Task Force, which includes members from Government, industry and academia, has raised several issues relating to security requirements relating to the Ada Run-time Environment. The ARTEWG Security Task Force coordinated the Orlando Security and Integrity - Ada Run-time Environment Workshop, sponsored by the Ada Joint Program Office (AJPO), IITRI and the University of Houston at Clear Lake City. The ARTEWG Security Task Force held a separate workshop in Baltimore late in 1990, reviewing the RRs, RIs and the first draft of the 9X requirements, and will be submitting a report to the Ada 9X Project Office in the next month addressing security concerns and issues.

Scope

Research is necessary to identify and define additional security requirements of Ada, as well as to assess the security impact of new proposed language features intended for the next Ada standard (1815B). Research should include Proof-of-Concepts, especially in the area of how the RTS relates to the TCB.

Assumptions

MCCR systems will increasingly need to use Ada in order to satisfy increasingly complex requirements, as well as adhering to the DoD Ada mandate.

Approach

The JLC should endorse and support the following activities:

- Focus research on the implementation of a TCB with an Ada RTS. Experience shows that this is currently not possible by configuring the Run Time Environment (RTE), but instead entails accessing the run-time source code and tailoring the RTE to meet the specific security requirements of the project. Fire-walling issues (e.g., should tasks be defined as the security subjects, or should programs be identified as the subjects) require further investigation and assessment. The results of this research and proof-of-concepts should be directed to Ada 9X to define the security requirements of MCCR systems for Ada 9X.

- Similarly, research is necessary to identify the constraints on the use of Ada (or specific Ada RTE features) to ensure various security requirements. Existing Ada language features, as well as proposed Ada 9X language features impacting security negatively must be identified early and avoided for security applications or modified appropriately by the language to minimize its impact on the security of MCCR applications.

- Research is also necessary in the application of formal verification techniques to the Ada language. Various "limited use" (safe subsets) of the Ada language are currently available and being studied. Additional research is necessary to extend the formal verification techniques to the entire language. Formal verification techniques exist for sequential programming languages (e.g., Floyd's and Hoare's methods); these are used to verify sequential (limited use) Ada. Research in the area of "Communication Closed Layers" and controlling non-determinism may aid in extending these formal verification techniques to the full concurrent Ada language set. The Ada 9X Project Office has tasked the Language Precision Team to formally define the Ada tasking model and minimize the implementation dependent features allowed in chapter 13 of the MIL-STD-1815A, "Ada Programming Language" Reference Manual.

2.4.8.7.2 Research Recommendation #2

- Develop tools and techniques for evaluations to better meet the needs of the DoD users.

Background

Evaluation is performed to provide information about security capabilities of a system. Evaluations are often costly, incomplete and require excessive resources. Evaluations take too long, are open to random interpretation, and produce results which do not meet the current needs of DoD users. The support for re-evaluation also needs to be addressed.

Scope

This research effort should focus on methods and tools for the timely and effective evaluation of products. These methods and tools should be aimed at evaluation consistency, timeliness, and maintainability.

Assumptions and Constraints

These tools should focus on basic product requirements that will provide meaningful input into the certification and accreditation process.

Approach

Tools and methodology need to be developed which standardize the evaluation process and produce timely results. Analysis of functions, features, and assurance should be automated to the extent possible to provide exception reporting that would point evaluators toward deficiencies. Tools that analyze calling trees, communication flow, use of global data, fault tolerance, software structure and modularity are required. Additionally, tools are required to map assurance evidence against requirements, specification and code. These tools should be developed to provide for increased efficiency of evaluators and provide meaningful input into the certification process. Various existing tools could be extended to meet the security needs described above.

2.4.8.7.3 Research Recommendation #3

- Provide for research to address the MCCR issues of high assurance that are not well understood today. These research issues include:

- Specification of Security Policy as it applies to high assurance.

- Architectural techniques for high assurance.

- Definition and broadening of the application of formal methods.

- Transition Formal verification out of TCSEC applications to broader applications.

Background

DOD 5200.28-STD, The Orange Book, was developed to address access control issues in DoD systems. A broader definition of security includes confidentiality, integrity, assured service, safety, data and process integrity, high reliability, mission criticality, and functional correctness. Specifying a security policy for confidentiality alone has undergone years of research, and although much progress has been made, it is still a topic of active debate. As one moves away from the more clearly-bounded, confidentiality-based definitions of security, the analyses and assurances become less formal and less rigorous because of the current technology lag in support of secure systems. Research is

needed in the specification of policies for systems whose security policy is broader than the better-understood confidentiality policies.

The methodology for defining the architecture of a TCSEC Trusted Computing Base is to partition the system into "trusted" and "untrusted" components. The trusted components implement the confidentiality policy, primarily access controls, and the untrusted ones do not. The concept is that the system can be structured such that greater design and other life cycle assurances can be applied to the trusted components. For systems implementing a non-TCSEC policy such as safety, there is no single methodology for structuring such a system to provide high assurances that the architecture implements the safety policy. Research into architectures to support such policies is needed.

A reasoning-based approach for developing secure systems provides a framework for the use of formal and rigorously-applied informal techniques to the system development process. A rigorous approach would provide a more sound, mathematical foundation for making assurance claims about the security of the system under examination. Since the types of systems that require broader notions of security are often very large, potentially involving millions of lines of code, formal methods must be selectively applied and scaled to these large efforts. Research into appropriate formal methods is needed, as well as more guidance on the appropriate role of formal methods in the development process.

One of the formal methods that can be used to assure system security is formal verification, although most of its past application has been limited to TCSEC applications where the security software is both very limited in size and concerned primarily with the confidentiality aspects of the TCSEC models. The experience gained here must be applied to other domains and enhanced methodologies, and robust and scalable tools must be developed.

Scope

This effort should focus on security requirements that are not addressed in the existing standards. These areas of research are applicable to those MCCR systems that require high levels of assurance.

Assumptions and Constraints

To meet the broad definition of MCCR requires that other issues be studied to effectively specify security requirements. It is difficult to specify requirements in an access control context where the system does not support users.

Long-term research will be required to provide the necessary breakthroughs to advance the technology for high assurance systems. As we rely more on computers to perform critical functions in MCCR environments the need for trust in their correct operation and their adherence to trust policy becomes more crucial.

Approach

Develop a framework for the definition of requirements of other disciplines such as process control and automata theory. This effort should help identify the set of requirements for building security into systems which are not satisfied by an access control policy. Security models should be integrated with other system models, such as those related to reliability and safety. New mechanisms should be developed to support critical aspects of integrity, distributed key management on low-security systems, availability, privacy assurance, and limitation on access in networks to permit interconnections of mutually suspicious organizations.

7.4.8.7.4 Research Recommendation #4

Perform research on cost benefit models for security.

Background

How much does security really cost and what are its real benefits? Accurate information on penetrations and loss of assets is often not available, and analyses must depend on expert opinion. Security is the minimization of risk through mechanisms, procedures, and rules. There is little information available which identifies the cost benefit model for information security. (See Garvey in the Bibliography for an example of this effort.)

Scope

Both the cost of production and the cost of use should be addressed. Benefit analysis must be based on careful risk analysis.

Assumptions and Constraints

In the DoD environment where the budget constraints mandate more effective use of system resources, the cost benefit model should identify areas where trust technology can increase effective use of resources.

Approach

Data collection and analysis of existing development activities such as DoD test beds could form the basis for evaluation of cost/benefit analysis. A reporting and tracking function may be required to facilitate model generation and validation.

2.4.8.7.5 Research Recommendation #5

Support research in secure distributed processing environments.

Background

The trust technology of today has evolved from the concepts of stand alone systems. The future will be based on distributed processing environments with heterogeneous components built into a system architecture. The definition of the architectural approach to distributed security must be developed to provide effective distributed processing services.

Scope

Research in this area should address architectural issues, global system requirements, interface standards, policy standards and implementation guidance to promote effective distributed processing.

Assumptions and Constraints

Most network protocol designs have tended to assure that networks will provide general interconnection. However, a common approach to achieving security in a distributed system is to partition the system into regions that are separated by a security perimeter.

Approach

Research is needed in the area of network protocols that allow partitioning for security purposes without sacrificing the advantages of general connectivity.

2.4.8.7.6 Research Recommendation #6

Establish new research programs and focus and coordinate existing research to identify a methodology for secure integration of systems composed of both trusted and untrusted heterogeneous components.

Background

The 1990's approach to MCCR system development has shifted from large-scale, special purpose software developments toward the use

of COTS, Government Off-The-Shelf (GOTS), and other reusable assets. New developments and development prototypes are using integrated, open architectures that minimize the amount of special purpose software that is needed. This trend may lead to important cost savings in the long run; however, there are many potentially high risk concerns that must be researched. In particular, the securing of systems composed of separately defined parts, even if all parts are trusted and evaluated, is not well understood. Theoretical research demonstrates that this effort is not trivial. Test bed experiments are on-going to test the integration of trusted products in various applications and varied environments. This important research needs to continue and must be coordinated to achieve maximum benefits.

The major issue is the need for an overall methodology for secure integration, and this critical research area must be investigated. Little is known about the system-wide security integration that is essential for a trusted MCCR computer environment. Defining what security means at the policy level and on design and implementation that follows is essential for accreditation of future systems. Therefore, there is a strong need for security engineering research to define a trusted integration methodology.

Scope

Research to define a secure integration methodology must be based on experimentation with actual products and assets within a testbed environment so that actual lessons learned and observations can support the theoretical aspects of the methodology. Research in the government and in industry must be coordinated, fostered and supported with a realistic goal for long-term success.

Assumptions and Constraints

The move toward the use of COTS, GOTS and other components is likely to increase as more powerful and flexible software products become available. The security implications of integrating these products into the future MCCR systems must be well understood to avoid potential security vulnerabilities in the operational systems. The growing need to reduce costs and to rapidly replace aging MCCR systems will also influence the COTS trend.

Approach

Research must be focused on lessons learned from test beds for COTS and GOTS integration and in prototypes and experiments with "plug in" components. The technology for securely integrating the components for a truly open architecture is still evolving. The near term research requires a focused view of current

security test bedded projects and a longer term task to derive a methodology for design, development and assurance for trusted system integration.

2.4.8.7.7 Research Recommendation #7

Focus computer research to incorporate security engineering support needs in the following areas:

- Intrusion Detection tools
- Security Audit Analysis tools
- Risk Analysis Methods and tools
- Process Management tools

Background

Research progress has been made in the application of artificial intelligence to computer tools. Intrusion detection tools for real time and off line monitoring of user and computer behavior and security audit tools for the management of large amounts of data, are needed to support secure MCCR operations. The security engineering process requires risk analysis methods and tools to assist with trade off decisions, design analysis, certification and accreditation. The risks involved in building trusted systems are addressed by prototypes, assessments, assurance activities and careful tracking and monitoring of the complex, concurrent project tasks. Research into support tools for configuration management and control to provide visibility and control for concurrent activities, to better evaluate project progress and to achieve consistency of notation is needed. Tools in these four areas will be important to facilitate security in a MCCR development.

Scope

The goal for application of artificial intelligence tools to security engineering must focus on leveraging existing prototypes and research in intrusion detection, in audit analysis and in risk analysis/risk assessment. For process management, process research, prototype research tools and tracking tools and commercial Computer Aided System Engineering (CASE) tools must be analyzed to determine applicability to support the secure software development process. Process programming is a new area of research, and the goals are more long term for the development of tools to support the engineering assurance and development of trusted system components.

Assumptions and Constraints

Research can be focused and refined in the security support area. Process programming for trusted system development will require long term funding.

Approach

Support research to determine feasibility of existing tools for intrusion detection, security audit analysis and risk analysis in the MCCR environment. Provide for additional research and development for the improvement of candidate tools or the creation of more or able tools. Select a best candidate tool to support risk analysis activities Provide for long-term research goals to explore secure system process representation, configuration management, CASE and tracking process management tools.

2.4.8.7.8 Research Recommendation #3

Establish research programs to define the requirements for and explore the feasibility of secure software engineering environments for the development of trusted MCCR systems.

Background

The environmental considerations for the development of MCCR systems requiring high assurance include all aspects of security with respect to personnel, physical environmental and procedural requirements. Additionally, the support tools used for system development, configuration management, security assurance and certification support must in turn be trusted to perform as intended. High assurance tools to support secure system developments are not currently available. Present practices do not ensure that the trustworthiness of the environment will match the required level of security for the MCCR system under development.

Scope

The focus of secure software engineering environment research must initially be on defining the requirements for environment security and later evolves to the applicability of specific tools (if any) that are a candidate for a secure environment. The secure integration of high assurance tools into the software engineering environment is an additional area to explore that is closely related to the research requirement to define a methodology for secure integration of COTS and GOTS components that comprise a MCCR system.

Assumptions and Constraints

Near term approaches for software engineering support assume most support tools are non malicious and will function as expected in a development environment, even one involving high trust assurance. This is necessary under the technology limitations of today.

Approach

Initiate a research effort to define trusted environment needs and to initiate requirements, policy and strategy for the long term goal of a trusted software engineering environment.

2.4.8.7.9 Research Recommendation #9

Foster reuse research for secure reusable assets within the MCCR application domain.

Background

Software asset reuse is a desirable goal for cost effective software developments of the future. Reusable assets will necessarily be broader than COTS, GOTS and software components alone. These assets must also include such elements as support documentation, software metrics, system analysis results, benchmarking results, assurance evidence and specifics about the application environment in which the asset was used. Reuse research is on-going and much work needs to be done.

In the specific area of reuse of trusted assets key issues include identification and development of an appropriate representation for a reuse library as well as the identification of associated asset information; determination of the impact of reuse of new, trusted components on an evolving system specification and analyses of the appropriate system structure for trusted systems in order to support and enable reuse; and determination of the trusted, reusable assets and their characteristics.

Scope

Trusted software reuse research must explore a variety of issues that are open and closely tied to the research areas of software reuse in general. Trusted reuse research results should be closely tied to research in trusted system maintenance. Many of the issues for precise clear documentation and support, strict, automated configuration control and assurance of asset integrity are essential for both trusted reuse and trusted maintenance. Domain analysis for reuse asset definitions must be conducted for trusted system developments.

Assumptions and Constraints

There are no MCCR reuse libraries (with support tools) defined or available for software reuse in the MCCR domain. Trusted assets must be defined for the MCCR reuse environment.

Approach

While it is ongoing in both government and industry, Trusted reuse research is still very new. Trust issues must be currently incorporated onto overall reuse research. Research in process models for system development that incorporate reuse and trust must be explored for MCCR reuse applicability.

2.4.9 CONCLUSION

The SA-I Computer Security / Software Integrity Panel (Panel IV) met 28 January through 1 February 1991 to discuss the issues discussed in this report and generate the JLC recommendations listed below. The panel members all agreed on the need to expand the definition of computer security (as defined in the TCSEC) to include software integrity and assurance in order to address the complex needs of MCCR applications, which are beyond those of basic ADP Systems.

The group noted that most of the recommendations of the Orlando II Security Panel were still valid and had not been addressed adequately. These recommendations were reviewed and the pertinent ones endorsed by the panel.

The panel identified the most relevant computer security software integrity issues and after much discussion and investigation, concluded with the following recommendations for the JLC:

- Establish policies and procedures for defining security requirements.
- Establish Certification and Accreditation policies and procedures.
- Identify system design and development disciplines aimed at achieving high assurance that the system's desired attributes (e.g., security, integrity, safety) are realized in the fielded systems.
- Identify operational policies and procedures ensuring the security and integrity of MCCR systems.
- Identify life cycle support - maintenance disciplines ensuring the security and integrity of MCCR systems.
- Identify and recommend specific educational needs that must be endorsed and supported.
    * Program Managers' Security Training Program
    * Life Cycle Security Requirements Course

    * Computer Security Awareness in Universities, Military Academies and other educational institutions

- Identify and recommend specific research needs that must be endorsed and supported.

    * Ada/Security Research

    * TCSEC Interpretations and Issues

    * Secure Integration of Varying Systems

    * Security Analysis Tools

    * CASE Security Tools

    * Formal Methods

    * Reuse

(INTENTIONAL BLANK)

ATTACHMENT IV-A
BIBLIOGRAPHY

Marshall Abrams, Diana Akers, Kathryn Bitting, Annabelle Lee, et al., "Computer Security for Acquisition Managers", MITRE Technical Report MTR-90W00138, MITRE Corp., McLean, VA, December 1990

Terry C. Vickers-Benzel, "Developing Trusted Systems using DOD-STD-2167A", in Proceedings of the Fifth Annual Computer Security Applications Conference, IEEE, December 1989.

Deborah J. Bodeau and Harriet G. Goldman, "Information Security in Security Acquisition", MITRE Report M89-46, MITRE Corp., August 1989.

Booz-Allen & Hamilton Inc., "Computer Security Answers for Acquisition Managers Course: Student Guide", BAH, 14 November 1988.

Department of Defense, "Trusted Computer System Evaluation Criteria", DOD 5200.28-STD, 26 December 1985.

Department of Defense, "Security Requirements for Automated Information Systems (AISs)", DOD 5200.28-STD, 21 March 1988.

Department of Defense, "Using the Department of Defense Trusted Computer System Evaluation Criteria in DoD Procurement (DRAFT)", National Computer Security Center, 6 November 1990.

J.N. Froscher, J.P. McDermott, C.N. Payne and H.O. Lubbes, "Successful Acquisition of Certifiable Applications Systems (or: How Not to Shake Hands with the Tar Baby)", in Proceedings of the Sixth Annual Computer Security Applications Conference, IEEE, December 1990.

P.R. Garvey, "A Framework for Estimating the Cost to Build Trusted Systems", in Proc. 24th Annual DoD Cost Analysis Symposium, Leesburg, VA, MITRE Corp., 1990.

Joint Logistics Commanders, "Proceedings of the Orlando II Software Workshop", Orlando, FL.

Logicon, Inc., "Program Managers Guide to Computer Security", Prepared for the National Security Agency under Contract N00039-85-D-0105, 31 March 1989.

Ann Marmor-Squires, Bonnie Danner, John McHugh, Lou Nagy, Dan Sterne, Martha Branstad and Pat Rougeau, "A Risk Driven Process Model for the Development of Trusted Systems", in Proceedings of the Fifth Annual Computer Security Applications Conference, IEEE, December 1989.

Rome Air Development Center, "SDS BMS Security Study Final Report: SDS Secure Development Methodology", Volume IV, TM-L-8361/025/00, Unisys, 11 October 1989.

TRW, "Process Model for High Performance Trusted Systems in Ada (For Development)", DARPA ACS Phase 1 Technical Report, TRW Inc., August 1989.

TRW, "Risk-Reduction Reasoning-Based Development Paradigm for Trusted Navy Command and Control Systems", STARS Final Report, 31 January 1991.

United States Air Force, "Guide For Security-Relevant Acquisitions", Volume 1, Air Force Cryptologic Support Center, Kelly AFB, TX, 1 May 1989.

United States Air Force, "Guide For Security-Relevant Acquisitions", Volume 2, Air Force Cryptologic Support Center, Kelly AFB, TX, 1 May 1989.

United States Air Force, "Computer Security in the Acquisition Life Cycle (DRAFT)", AFSSM 5010, Air Force Cryptologic Support Center, Kelly AFB, TX, June 1990.

United States Air Force, "Complex System Guidelines (DRAFT)", Air Force Cryptologic Support Center, Kelly AFB, TX, 28 September 1990.

United States Air Force, "Security Policy Generation Guide (DRAFT)", AFSSI 5001, Air Force Cryptologic Support Center, Kelly AFB, TX, 21 December 1990.

United States Navy, "Computer Security Guidebook for Mission-Critical Computer Resources Managed Under the Research, Development, and Acquisition Process", Space and Naval Warfare Systems Command, 26 October 1990.

ARTEWG Security Task Force, "Orlando Security and Integrity Issues for the Ada RunTime Environment Workshop - March 1990," SIGAda Ada Letters, November/December 1990.

F. Maymir-Ducharme, "Combining Ada RTE and Security Issues, ARTEWG Security Task Force Investigations," Proceedings of the 13th National Computer Security Conference, Washington, DC, 1990.

National Computer Security Center, "Embedded Real-Time Trusted Computing System Requirements for Development, Operational and Maintenance Environments," prepared by IITRI, June 1990.

National Computer Security Center, "Extending the ERT-TCS System Requirements to the TCSEC," prepared by IITRI, August 1990.

National Computer Security Center, "Issues Building a Trusted Computing Base (TCB) at the B3 Level in Ada," prepared by IITRI, August 1990.

National Computer Security Center, "Information Security Products and Services Catalogue," Fort Meade, MD. (no date)

National Academy Press, "Computers at Risk Safe Computing in the Information Age," 1991.

(INTENTIONAL BLANK)

2.5 PANEL V FINAL REPORT: SOFTWARE CONFIGURATION MANAGEMENT

2.5.1 INTRODUCTION

This report addresses the efforts of Panel V toward addressing CM issues.

2.5.2 OBJECTIVE

Identify potential MCCR acquisition and support problems relative to implementing CM for software, identify issues and recommend solutions to problems/issues.

2.5.3 BACKGROUND

The Defense Quality Standardization Office (DQSO) and the JLC JPCG-CRM PDSS Subgroup are jointly sponsoring the development of a military standard and associated handbook covering all aspects of Defense CM. This project, which was initiated as a result of Orlando II recommendations, is a critical element of the DoD program to reduce the number of military standards, specifications, and related documents to a manageable level by eliminating redundancies, discrepancies, inconsistencies, and outdated material. The project will provide, for the first time, a comprehensive, top-down approach to CM as it relates to all aspects of DoD weapon system acquisition and life cycle support, including hardware, firmware, and software configuration items. Although two new documents will be developed, eleven existing documents will be canceled and deleted from the Government standards inventory. Several documents that may be canceled include MIL-STD-480B, "Configuration Control - Engineering Changes, Deviation, and Waivers", MIL-STD-481B, "Configuration Control - Engineering Changes (Short Form) Deviations and Waivers", MIL-STD-482A, "Configuration Status Accounting Data Elements and Related Features", MIL-STD-483A, "Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs" and MIL-S-83490, "Specifications, Types and Forms". Additional objectives of the project are to identify conflicts and other problem areas among the current versions of associated standards as they relate to CM, particularly MIL-STD-490A, "Specification Practices", MIL-STD-499,"Engineering Management", and DOD-STD-2167A, and to prepare formal recommendations for resolving these problems. MIL-STD-973 is scheduled to be published in late summer 1991. The 25 January 1991 draft version of MIL-STD-973 was reviewed by Panel V.

2.5.4 SCOPE

Although the scope of the panel's efforts was essentially unbounded, they were given the responsibility for accomplishing the following tasks with the limitation that comments and recommendations should focus on software CM:

- Review existing draft CM standard and other documents. Identify areas of conflict or incompatibility. Identify issues and provide proposed recommendations for resolving problem areas.
- Develop recommendations for implementing guidance for transitioning from existing CM standards to the new standard.
- Review CM status accounting data elements. Develop list of minimum basic set of CM data elements.
- Investigate area of CM-CASE environment logical interfaces. Develop recommendations pertaining to standardizing these interfaces.

The panel was to generate the following products as a result of completing the tasks:

- Technical report based on review of CM and related standards, including list of conflicts and recommended changes. (Task 1)
- List of CM standard implementation guidelines. (Task 2)
- List of basic CM data elements. (Task 3)
- Technical report on CM-CASE logical interfaces, including list of recommendations for standardizing these interfaces. (Task 4)

2.5.5 ASSUMPTIONS and CONSTRAINTS

- The recommendations of the panel are to be submitted for review by the PDSS Subgroup/CM Advisory Group (CMAG).
- Final determination of the content of MIL-STD-973 rests with the PDSS Subgroup/CMAG.
- Only a preliminary review of the draft standard was feasible in that the draft was not available for review until the workshop began (28 January 1991).

- There are constraints on changing the contents of MIL-STD-973 to conflict with DOD-DIR-5000.1 and DOD-INST-5000.2.

2.5.6 APPROACH

- Review tasks as a group to list potential conflicts.
- Select a government/industry team for each task.
- Define the problem and examine characteristics.
- Identify solutions and alternatives.
- Select best solution and alternative.
- Submit for consensus.

2.5.6.1 The panel listed their concerns for each task. The tasks and corresponding concerns are listed as follow:

- Task 1: Review existing draft CM standards and other documents. Identify areas of conflict or incompatibility. Identify issues and provide proposed recommendations for resolving problem areas.

Concerns:

Are hardware/software/system requirements covered and specified where unique?

Is CM addressed as an organization, or as a discipline?

Does the standard address all types of software?

Does the standard allow for a quick reaction change?

Does the standard establish Government tasking?

Are firmware and software adequately addressed?

Does the standard include inappropriate (for a standard) Government policy?

Does the standard address all appropriate CM standards, including DOD-STD-1679, "Software Development"?

Is each paragraph stated as a requirement?

Does the standard provide CM coverage of NDI and COTS?

Should software be excluded from Physical Configuration Audit (PCA)?

What coverage is provided for management of software technical data by CM?

Are unique requirements for software and hardware configuration change control addressed?

Is CM of software documents as well as hardware documents addressed?

Is CM of Management Information System (MIS) discussed?

Is the compatibility between formal baselines and developmental configuration addressed?

Are the definitions of baselines correct?

Is the identification of CSCIs addressed adequately?

Is adequate coverage of PDSS provided?

Is the standard compatible with Ada?

Is the standard compatible with system, hardware, and software prototyping? (Alternate Life Cycles)

Is the standard compatible with CALS?

Are there unnecessary constraints to CM automation?

Is CM's role in risk management addressed?

- Task II: Develop recommendations for implementing guidance for transitioning from existing CM standards to the new standard.

Concerns:

Is there a need for formal training for the new standard?

What guidance is required for the use of the new standard on existing contracts?

Is there a need to endorse the development of a CM handbook to support implementation of MIL-STD-973?

There is a need to ensure that appropriate policy statements deleted from the draft standard are incorporated in DOD-INST-5000.2.

Have CM requirements been changed from the current CM standards in MIL-STD-973?

A matrix should be furnished to provide traceability of requirements between the superseded CM standards and MIL-STD-973.

Are evaluation criteria provided to aid in the transition to the new CM standard. (Hardware or Software)

Are incentives for the government Program Management Office required to promote the use of the new CM standard?

- Task III: Review CM Status accounting data elements. Develop a list of the minimum basic set of CM data elements.

Concerns:

Address the relationship and potential for conflict with CALS data elements.

Do the selected data elements provide traceability for software problem reports. (SPR #s, PCR #s)

Are software development and support elements defined?

Should CM data elements be addressed in MIL-STD-973 or in MIL-STD-482?

What status accounting data element categories are required?

Are required levels of detail provided?

What is the relationship of Task III to Task IV.

- Task IV: Investigate area of CM-CASE environment logical interfaces. Develop recommendations for standardizing these interfaces.

Concerns:

What is the relationship of Task IV to Task III?

Is CM of software which is not a CI addressed?

Is the new CM standard compatible with DOD-STD-1467 "Software Support Environment", and DOD-STD-2167A

Is classification of various levels or types of software addressed?

Within the CASE environment, is CM of tools addressed?

2.5.6.2 DISCUSSION OF DEFINITIONS:

Need Revising

3.4
3.5
3.5.1
3.5.2
3.5.3
3.6
3.8
3.9
3.10
3.11
3.12
3.14
3.15
3.17
3.17.1
3.17.2
3.17.3
3.19
3.20
3.21
3.23
3.27
3.28
3.29
3.38
3.39
3.41
3.45
3.47
3.49.1
3.54

Add New Items

1. Acronyms: CSCI, Computer Software Component (CSU), (CSC)
2. Software Definition
3. Commercial Off-the-Shelf (COTS)
4. Non-Developmental Software (NDS)
5. P/CR
6. Contracting Officer's Technical Rep.(COTR)
7. Segment
8. Version
9. Class I & II
10. SCO
11. Manufacturing Review Board (MRB)
12. Developmental Configuration
13. Formal Design Reviews Software Requirements Review (SRR), Software Specification Review (SSR), System Design Review (SDR), PDR
14. Authentication

Need Revising

3.57
3.61
3.67
3.70
3.72
3.73
3.74
3.76
3.79
3.80

2.5.6.3 DESIGNATION OF SUBTASK MEMBERS

Task I:	Dennis E. Nickle
	Johnnye Burnham
	Ron Pruiett
	Linda Burgher
	Bruce Dubbs
Task II:	Larry Griggs
	Jerry Raveling
Tasks III & IV:	Chuck Kuniyoshi
	Richard Seyfert
	A. Brett Pope

2.5.7 DETAILED REPORT

The recommendations discussed in the Executive Summary reflect the concerns of practicing CM professionals from the government DoD components and the defense industry. These concerns reflect the panel's desire that MIL-STD-973 addresses both hardware and software requirements in any given environment and that these requirements are stated in contracts issued by the Government to defense contractors and by one DoD component to another DoD component.

A specific issue with this panel was the application of MIL-STD-973 to a software project, to a project where software is integral to the hardware, or to a project where the software drives the hardware. This desire is reflected in some of the 15 plus recommendations for inclusion of software terminology in specific paragraphs, which on the surface may appear to be non-essential, but are critical to both the Government's Administrative Contracting Officer (ACO) and the defense contractor in terms of assuring that the contractual requirements are adhered to. For example the recommendation to change a portion of paragraph 3.6 from "For CIs, an..." to "For hardware CIs, an..." will provide a clarification of this definition and prevent misunderstandings which are so prevalent in contractual relations today.

The reference to a handbook currently numbered MIL-HDBK-61, "Configuration Management" (Draft) is most important to the successful implementation of MIL-STD-973. Because of the amalgamation of several standards, it will be difficult enough for contractors and government procurement activities alike to realize that there is no longer a MIL-STD-480B or a MIL-STD-1521B - for the functional and physical configuration audits. Therefore, a well-written guide for applying this standard will be most important.

The CMAG must consider, and we believe that this is consistent with MIL-STD-962, "Military Standards, Handbooks, and Bulletins; Preparation of", that the several appendices in MIL-STD-973 be converted into DIDs for the standard. In addition, the CMAG must consider the utilization of DIDs for the reduction of Section 6 which, at present, is felt to be too long to the point that pertinent points will be lost or will not be apparent to the user.

Some of the major recommendations that must also be considered by the CMAG and the PDSS Subgroup include that, if CI is used for hardware and software, that provision be made for software specific situations in which one can declare a CSCI.

The term operational baseline is offered, especially for the PDSS environment, to provide a means of establishing the current configuration identification of a system and its CIs. Along with this is an important recommendation that defines the level of control for software which shall be applied at the binary code level.

The requirements of software developmental CM and the non-conformance corrective action process are also important in meeting the software developmental configuration management requirement. Pursuant to the requirement of Task 2 for implementation guidance, the theme here will be to apply a consolidated set of standards combined into MIL-STD-973 for the performance of the CM process for software. Thus provisions for a supersession matrix, guidance to the contracting agency, and use of the new MIL-STD-499 and MIL-HDBK-61, along with training criteria and the associated transition criteria, will all be important when MIL-STD-973 is first invoked.

The panel did not have the resources to fully provide a status accounting data elements list, but has listed those items that can now be defined at conferences such as the May CM/Data Management (DM) Conference or by such groups as the Electronic Industries Association's (EIA's) G-34 Computer Resources Committee.

Finally the task relating to CM-CASE tools was too nebulous to be fully addressed and, suffice it to say, that all tools normally

under contractor configuration control, or that of the software support activity, shall remain under such control for the life of the software product.

The panel discussed issues that arose during their review of MIL-STD-973. Their primary concerns along with consensus and rationale are listed as follows:

- Should an ECP Form for software be included?

  * Consensus: Modify the existing ECP Form by attaching a page 1S and page 2S with software terms and semantics.

  * Rationale: This will, for once and for all, eliminate the confusion of using a hardware oriented form to request and justify changes to software.

- What is meant by the physical characteristics of software?

  * Consensus: The physical characteristics of software are its binary image. Whenever the binary image changes, it constitutes a change to the physical characteristics of the CSCI. Anything in the Software Engineering Environment (SEE) or Software Test Environment (STE) which is necessary to re-generate or to engineer the binary image of the CSCI is a "critical component" in the SEE or STE.

  * Rationale: This convention provides a uniform, easy-to-understand, and useful definition of the physical characteristics of software.

- What should be the level of configuration control for software?

  * Consensus: The critical level of configuration control should be the binary image (or object code) which is the deliverable product.

  * Rationale: This emphasis extends configuration control to all items affecting the binary image, i.e., source code and the critical components of the SEE and test environment.

- Should CSCI be used where software specific reference is needed?

  * Consensus: The Scope section shall state that CI is preferred except where a software specific term is needed. In this case, CSCI will be used.

  * Rationale: This will eliminate the need for terms such as HWCI or CSCI. A software-specific term is occasionally required for clarity. In this case, DOD-STD-2167A provides the accepted term and definition.

- Can there or should there be more than the 3 formal baselines?

  * Consensus: Yes. Operational Baseline may be established during PDSS.

  * Rationale: The Operational Baseline allows for final adjustment after Functional System Audit (FSA)/Formal Qualification Review (FQR) test bed environment to present a clean baseline into PDSS. This also eliminates the term, current configuration identification.

- Is the software reference in the engineering release subparagraph necessary in the status accounting section?

  * Consensus: No. Delete the software reference in Appendix G of DOD-STD-2167A and refer to the VDD for release of software in the status accounting section of MIL-STD-973.

  * Rationale: A software release is completely different from hardware - it's better to keep the two elements separate.

- Should software documentation be addressed?

  * Consensus: The standard should reference or define software documents as well as hardware. Hardware should be referenced to MIL-STD-490A and software should be referenced to DOD-STD-2167A.

  * Rationale: Avoids confusion, especially with MIL-STD-490A in its present state.

- How does Block Change Procedure as presently defined affect software?

  * Consensus: Keep the current definition.

  * Rationale: The current definition still applies. EE/EA Programmable Read-only Memory (PROMS), PALS, etc., come under provisions of DOD-STD-2167A for such programmable software.

The panel concluded that MIL-STD-973 needs to contain guidance for CM for PDSS. Currently, every organization seems to have their own unique procedures for CM. If the process is to be standardized, guidance is required for the PDSS community. A handbook would be helpful for the PDSS effort.

2.5.8 RECOMMENDATIONS

Factors

Time Factor: All of these recommendations are for near-term solutions.

Return on Investment (ROI): The ROI is hard to determine.

Estimated Cost and Time to Implement: Unknown.

Dependencies include:

- CMAG or PDSS Subgroup committee actions.
- The timely issuance of MIL-STD-973 (by September 1991).
- The issuance of a revised MIL-STD-499.
- The issuance of a handbook, e.g., MIL-HDBK-61.

Alternatives:

- No CM standard.
- No accomplishment of stated objectives.

Method of Implementation: See Recommendation 05-05-06.

Detailed Products: A comprehensive standard covering hardware/software requirements.

Panel Recommendations.

2.5.8.1 Recommendation 05-05-01

Addition to Paragraph 3.18.

- Recommendation: Recommend the following be added to paragraph 3.18 of MIL-STD-973 to clarify that CSCIs are referred to as CIs:

  * "In this document, the term CI is used for literary purposes, and may be interpreted as either a hardware CI or a Computer Software CI, CSCI, as appropriate."

2.5.8.2 Recommendation 05-05-02

Add New Paragraph 3.5.4.

- Recommendation: Add the following new paragraph to MIL-STD-973:

  * "3.5.4 Operational Baseline. The then current approved documentation describing all of the necessary functional and physical characteristics of the system and its CIs, and any required joint and combined operations interoperability characteristics of the system and its CIs. The Operational Baseline may be established by the Government during PDSS.

- Rationale: Use of the terminology, Operational Baseline, is common in the PDSS environment. The Operational Baseline designator provides a means of establishing the then current configuration identification of a system, and its CIs as implemented in the operational environment. Use of Product Baseline (PBL) does not appear to be appropriate for the operational system.

  * PBL establishes the "build to" design for hardware and the "as built" design; this baseline designation is germane to the acquisition of the system, and its CIs.

  * Change is our reality in the operational environment. Thus, the configuration of the system and its CIs are constantly being modified to meet new operational requirements. Use of Operational Baseline provides a more accurate representation of the nature of the operational baseline.

2.5.8.3 Recommendation 05-05-03

Engineering Change Proposal Form.

- Conflict: Most items in the existing ECP Form shown in MIL-STD-973 are not applicable to CSCIs.

- Recommendation: Include the Software Change Proposal (SCP) Form from DOD-STD-1679A and include it in MIL-STD-973 as an attachment to the ECP Form for software only changes and for software portion(s) of system/hardware/software changes.

2.5.8.4 Recommendation 05-05-04

Level of Configuration Control for Software.

- Conflict: MIL-STD-973 does not clearly define the level (i.e., product specification and source listing, source code, object code, etc.) that configuration control is applied to computer software.

- Recommendation: Include a requirement in MIL-STD-973 that configuration control shall be applied to computer software at the binary image level, and that all tools and documentation used to generate a binary image of the computer software shall be under configuration management.

2.5.8.5 Recommendation 05-05-05

Add paragraphs 4.9 and 4.10.

- Recommendation: Add the following paragraphs to MIL-STD-973:

  * "4.9 Software Developmental Configuration Management

    The contractor shall implement, as part of the project's configuration management program, configuration management of the evolving software's developmental configuration. Software developmental configuration management consists of the following elements; configuration identification, control, and status accounting.

    4.10 Non-Conformance Corrective Action Process

    The contractor shall document and implement a corrective action process for handling all

problems detected in the products under configuration control; both formally baselined and developmental configuration products. The process shall utilize problem/change reports to describe each problem detected in the system, software, hardware, or firmware or documentation that has been placed under configuration control."

- Rationale: The following rationale is provided for the inclusion of the requirement for Software Developmental CM and the Non-Conformance Corrective Action Process in MIL-STD-973 as General (Section 4) and Detailed (Section 5) Requirements:

  * Software Developmental CM

  * A principal CM feature of DOD-STD-2167A is the control of the evolving configuration of the software design and implementation. Software developmental CM is under the development contractor's, versus the Government's, control. Software developmental CM is implemented during the Preliminary Design phase and continues through the balance of the development phases until the formal Product Baseline is established. Developmental CM serves to "fill the gap" between the establishment of the formal Allocated Baseline and the formal Product Baseline. Without the requirement for the contractor to implement Developmental CM, a significant increase in risk and potential loss of project management control is experienced.

  * For purposes of compatibility between MIL-STD-973 and DOD-STD-2167A, and standard software developmental CM practices, it is mandatory that the requirement for software developmental CM be included. Further, it should be noted that modern computer system hardware development is evolving towards an approach which will, if it has not already, require a developmental CM approach.

  * Non-Conformance Corrective Action Process

  * Critical to effective CM of system software is the establishment of a Non-Conformance Corrective Action Process which is required to control all products under configuration control, both formal and developmental configuration. DOD-STD-2167A, Section 4.1.9, establishes the requirement for the contractor to establish a corrective action process. The process, similar to formal CM

control, serves to document a problem, review and analyze the problem, implement (where required) corrective action, test/verify the correction and release the problem correction.

* For purposes of compatibility between MIL-STD-973 and DOD-STD-2167A, and standard software development and PDSS CM, it is mandatory that the requirement for a Non-Conformance Corrective Action Process be included.

2.5.8.6 Recommendation 05-05-06

MIL-STD-973 Implementation Guidance.

- Recommendation: The DoD (DQSO), and/or the services (as appropriate) should issue implementation guidance for MIL-STD-973 concurrent with its publication. The implementation guidance should address, as a minimum, the following: (Information is presented in suggested letter format.)

* "1. Synopsis. MIL-STD-973 is the single, overall standard within the DoD for CM of systems, hardware, software, firmware, and manufacturing processes. It applies the four elements of CM - configuration identification, configuration change control, configuration status accounting, and configuration audits - to systems, hardware, software, firmware, and manufacturing processes. MIL-STD-973 does not prescribe any new CM requirements. Instead, it includes all existing general CM requirements from the superseded DoD and military standards listed below. Unique CM requirements for systems, hardware, software, firmware, and manufacturing processes are also identified.

2. Supersession. MIL-STD-973 supersedes the following DoD and military standards:

(1) MIL-STD-480B

(2) MIL-STD-481B

(3) MIL-STD-482A

(4) MIL-STD-483A

(5) MIL-STD-1456

(6) MIL-STD-1464

(7) MIL-STD-1662B

(8) MIL-STD-2140

(9) MIL-STD-83490

MIL-STD-973 supersedes the CM portions of MIL-STD-1521B, which include:

(1) Paragraph 3.7 - FCA

(2) Paragraph 3.8 - PCA

(3) Paragraph 5.1.7 - FCA

(4) Paragraph 5.1.8 - PCA

(5) Appendix G - FCA

(6) Appendix H - PCA

3. Contracting Agency Use of MIL-STD-973.

a. New Contracts. On and after (date), all DoD contracting agencies which include CM requirements within each Request for Proposal (RFP) and/or Request for Quote (RFQ) for a new contract shall invoke MIL-STD-973.

b. Contract Modifications. On and after (date), all DoD contracting agencies which include CM requirements within each RFP and/or RFQ for a contract modification should invoke MIL-STD-973 whenever:

(1) RFP and/or RFQ required for development of new Configuration Items (CIs).

(2) RFP and/or RFQ required for significant enhancements to existing CIs.

4. Contracting Agency Use of MIL-HDBK-61. MIL-HDBK-61 contains recommendations for invocation, tailoring, and implementation of MIL-STD-973. Also included within MIL-HDBK-61 are detailed traceability matrices between MIL-STD-973 and all DoD and military standards superseded by MIL-STD-973.

5. Related Standards.

a. DOD-STD-2167A. There are tentative plans by DoD to remove, within the next several years, basic software cm requirements data from DOD-STD-2167A. Until the change is implemented, for software, it shall be necessary for the Government to cite both DOD-STD-2167A and MIL-STD-973 in contractual documents to properly implement CM requirements.

b. MIL-STD-1521B. With the publication of MIL-STD-973, the FCA/PCA appendices of MIL-STD-1521B should be tailored out since these appendices are now in MIL-STD-973."

2.5.8.7 Recommendation 05-05-07

Use of MIL-STD-499A (USAF).

- Problem: MIL-STD-499A (USAF) is in the process of being revised to include all of MIL-STD-1521B (less its Functional Configuration Audit (FCA) and PCA provisions which are included in DRAFT MIL-STD-973). However, as of 1 February 91, the MIL-STD-499 revision has not been issued. This causes problems with the invocation of MIL-STD-499 within DRAFT MIL-STD-973 because the current MIL-STD-499A (USAF) does not include any requirement for technical reviews (SSR, PDR, CDR, etc.). This situation will cause problems during service review of MIL-STD-973.

- Recommendation: Wherever MIL-STD-973 specifies requirements related to technical reviews (SSR, PDR, CDR, etc.), invoke both MIL-STD-499 (USAF) and MIL-STD-1521B until MIL-STD-1521B is superseded by MIL-STD-499A.

2.5.8.8 Recommendation 05-05-08

MIL-HDBK-61, Guide for Application and Tailoring of Configuration Management Requirements.

- Recommendation: Recommend that DoD (DQSO) remain committed to the development and publication of MIL-HDBK-61. The handbook should provide:

  * Implementing guidance for the application of MIL-STD-973 for CM of systems, hardware, software, firmware, and the manufacturing processes.

* Tailoring guidance on the application of MIL-STD-973, to include:

  Transfer of data currently found in Section 6.2, Tailoring Guidance for Contractual Application, to MIL-HDBK-61.

  Application of CM to less than major defense systems, MIS, system or system component prototypes, hybrid systems (i.e., combination of custom built, NDI, and reusable components), non-deliverable software, and for PDSS.

  Traceability matrix which provides:

    Traceability of CM requirements derived from those documents listed in Section 6.2 of MIL-STD-973, Supersession Data, to the requirements sections (Sections 4 and 5) of MIL-STD-973.

    Traceability of requirements cited in Section 4, General Requirements, to those cited in Section 5, Detailed Requirements, of MIL-STD-973.

    Traceability of requirements cited in MIL-STD-973, Sections 4 & 5, to corresponding paragraphs, when appropriate, in MIL-HDBK-61.

2.5.8.9 Recommendation 05-05-09

Recommend the following be included in MIL-STD-973 as a minimum set of CM status accounting data elements:

Priority
Software Control Number/Version
Patch Control Number
SCN Control Number
Associated Change Request
CR Control Number
CR Status and Date
User Activity
- Organization
- Initiator
- Telephone Number
- Date Incident Occurred
- Date
- Time

Problem Category (Documentation/Code)

Executable Code Address
Problem Duplicated
Test Specification Reference
Description of Incidence (Text)
Developing Activity
Problem Analysis - Title
For ECP:
- Application Level (System, Hardware, Software, Documentation, Other:
- CSCI
- CSCI Affected
- Programming Tape (media)
- Development System
- Software Tools
- Software Tool Development
- Development Hardware
- Target Hardware
- Operational Support Facility (OSF) Modification Cost

Software Security Classification (C/U/S/T - level of classification and why)
Software Subsystem/Tool
Medium Type (Unlabeled/labeled/hidden/low density)
Special Instructions (Text)
Tape/Disk (X of Y)
Software Version
Software Revision Level
File Number
File Revision Level
File Name
Link Number
File Security Classification (U/S/T/SECRETNOFORN)
Development System
Projected Delivery Date
Actual Delivery Date
Firmware Number
Firmware Revision Level
Contractor Comments (Text)
Subroutine Name
Subroutine Revision Level
Subroutine Security Classification
Processing Time
Memory
Access Time
Disk Space
Special Instructions (Text)
Development System (CASE/hardware/other software)
Code - Language
Design - Language
System/Project Name
Problem Title
Problem Description

Analyst Name
Date Analyst Assigned
Data Analysis Complete
Recommended Solution
Impact (of Change Action)
Corrector's Name
Follow-up Disposition
Approval
Contract Number
Time to Analyze
Time to Correct
Close Out Date
Other Software Elements Affected
Implementation Solution

2.5.8.10 Recommendation 05-05-10

Add New Paragraphs 4.11 and 4.12.

- Recommendation: Add the following paragraphs to MIL-STD-973:

  * "4.11 Non-Developmental Item (NDI)/Privately Developed Item (PDI).

    The contractor shall recommend the methods to be implemented for configuration management of a Non-Developmental Item (NDI) and a Privately Developed Item (PDI), where required. The basic and detailed requirements of this standard must be tailored to reflect the unique CM requirements and methods required to effectively manage the configuration of:

    a. COTS hardware, software, and firmware products available from the commercial market place which are to be utilized for a specific program with or without modification to the COTS established configuration.

    b. Inheritable hardware, software, and firmware products available within the government inventory which are capable of fulfilling DoD requirements.

    c. Privately Developed Item (PDI) (see paragraph 3.59)

    Reference MIL-HDBK-61 for guidance on tailoring of this standard for NDI/PDI.

4.12 Prototype CM

The contractor shall recommend methods to be implemented for configuration management of all hardware, software, and firmware prototypes. The methods should consider the intended use of the prototype (e.g., throw-away or evolving product), phase of development, and the required level of government and development contractor control of the prototype."

2.5.8.11 Recommendation 05-05-11

Training.

- Problem: No special training requirements specific to the implementation of MIL-STD-973 were identified. MIL-HDBK-973 does not represent, in the opinion of the Task 2 Subpanel, the levying of new or unique (to current practices) CM requirements.

- Recommendation: MIL-HDBK-61 should provide a training criteria guideline for the application of MIL-STD-973.

2.5.8.12 Recommendation 05-05-12

Transition Criteria.

- Recommendation: Recommend that MIL-HDBK-61 provide guidance to the Program Manager and/or Configuration Manager in the form of a set of criteria that can be used to determine, for a particular project, the benefits, and/or problems inherent to transitioning the project to MIL-STD-973 during deployment and/or support.

2.5.8.13 Recommendation 05-05-13

Policy Statements.

- Recommendation: Recommend that policy statements edited from the reviewed draft of MIL-STD-973 should be forwarded to DoD (DQSO) for consideration for inclusion in DOD-INST-5000.2 or DOD-INST-5000.4, "Mission-Critical Computer Resources (MCCR) Program".

2.5.8.14 Recommendation 05-05-14

CM of Development Environment.

- Recommendation: MIL-STD-973 must require that the development environment, including the SEE/STE (COTS,

CASE, and contractor proprietary software, etc.) must be maintained under contractor internal CM for the life of the contract or until PMRT. This requirement can easily be appended to paragraph 4.4.

- Rationale: There is a requirement for configuration identification and change control of all environmental software (COTS, CASE, NDI, PDI, etc.) from the initial point of contract award to ensure the procuring activity can provide a maintenance environment.

2.5.8.15 Recommendation 05-05-15

Change to MIL-STD-973, Appendix G.

- Recommendation: Change "...shall contain the standard..." in paragraph 40.1.2 to "...shall contain directly, or by reference, the standard...".

2.5.8.16 Recommendation 05-05-16

Change to MIL-STD-973, page 55.

- Recommendation: Delete "NOTE: Minor clarifications...by the procuring activity." from the middle of page 55.

2.5.8.17 Recommendation 05-05-17

Change to MIL-STD-973, page 64.

- Recommendation: Change the 2nd and 3rd lines on page 64 from "...into the Product Configuration Identification (PCI). Class II changes are applicable during production and are normally identified..." to "...into the CI. Class II changes are normally identified..."

2.5.8.18 Recommendation 05-05-18

Change to MIL-STD-973, page ii, FOREWORD.

- Recommendation: Renumber paragraph 4 on page ii as paragraph 5 and insert a new paragraph 4 as follows:

  * "4. This standard is not intended to specify or discourage the use of any particular configuration management method. The contractor is responsible for selecting the configuration management method (for example, an automated change proposal tracking system) that best supports the achievement of contract requirements."

2.5.8.19 Recommendation 05-05-19

Restructuring Format of MIL-STD-973.

- Recommendation: Recommend MIL-STD-973 be restructured as follows:

iii		Add Aircraft Configuration Control Board (ACCB) in Section 3. Move Section 3 to an Appendix.
v	4.2.1	Omit subcontractor statement.
	4.3	Change "---planning" to "--- process".
	4.4.2	Make this 4.3.1 under Process.
	4.5	Make this 4.3.2 under Process.
	4.7.1	Omit. Covered in Section 5.
	5.2	This paragraph related to 4.3. Change "---planning" to "--- process".
	5.2.6	Change to Interface Management Process
vi	5.2.7	DM deserves full treatment - put in separate document or expand total of MIL-STD-973 to CM/DM.
vi	5.3	Config I.D.
	5.3.2.3.1	Put in a DM standard.
	5.3.3	Put in a DM standard.
	5.3.3.1	Put in a DM standard.
	5.3.3.2	Put in a DM Standard.
	5.3.3.3	Put in a DM standard.
	5.3.3.4	Put in a DM standard.
	5.3.3.5	Put in a DM standard.
	5.3.3.6	Put in a DM standard.
	5.3.5	Put in a DM standard.
	5.3.6	Put in special Evaluation & Validation Section.
	5.4	Related to 4.5. Change to 5.2.6.1.
	5.5.1.3/	Contractor/contract requirements.
	5.5.2.2/	- change so all covered by one entry
	5.5.3.1	
vi/vii	5.5.2.5/	Post-actions. Write once.
	5.5.3.4/	
	5.5.4.6	
	5.6.4	Change to ECPs. Create 5.6.4.1 to cover this.
	5.6.5	Change to 5.6.4.2.
vii	5.6.6	Change to 5.6.4.3.

	5.6.7 thru 5.6.12	Change to 5.6.4.4 thru 5.6.4.9.
	5.6.14	Change to 5.6.4.10.
	5.6.14.1 thru 5.6.14.1.2	Change to 5.6.4.11 thru 5.6.4.13.
	5.6.15	Change to 5.6.4.14.
	5.6.16	Cover under Process.
	5.6.17	Change to 5.6.4.15.
	5.6.18	Make this 5.6.5.
	5.6.18.1 and 5.6.18.2	Make this 5.6.5.1 & 5.6.5.2.
	5.6.18.3	Cover under Process.
viii	5.6.18.4	Make this 5.6.6.
	5.6.18.4.1 thru 5.6.18.4.9	Make this 5.6.6.1 thru 5.6.6.9.
	5.6.18.5	Change Title to "TYPES OF ECP" and make this 5.6.7.
	5.6.18.5.1	Change to 5.6.7.1.
	5.6.18.5.1.1	Cover under Process
	5.6.18.5.2	Change to 5.6.7.2.
	5.6.18.6	Change to "ECP PRIORITY TYPES" and make this 5.6.8.
	5.6.18.6.1 thru 5.6.18.6.3	Make this 5.6.8.1 thru 5.6.8.3.
	5.6.18.7	Make this 5.6.4.15.
	5.6.18.7.1	Make this 5.6.4.15.1.
	5.6.18.7.2	Make this 5.6.4.15.2.
	5.6.18.8 thru 5.6.18.10	Make this 5.6.4.15.3 thru 5.6.4.15.5.
	5.6.18.11	Cover under Process.
	5.6.19	Change to 5.6.8 Class II ECPs.
	5.6.19.1 thru 5.6.19.5	Make this 5.6.8.1 thru 5.6.8.12.
	5.6.20	Make this 5.6.9.
	5.6.21 thru 5.6.24	Cover under Process.
	5.6.25	Deviations. Change to 5.6.10.
viii/ix	5.6.25.1 thru 5.6.25.8	Make this 5.6.10.1 thru 5.6.10.13.
ix	5.6.26	Waivers. Change to 5.6.11.
	5.6.26.1 thru 5.6.26.8	Make this 5.6.11.1 thru 5.6.11.13.

5.6.27	Not Operationally Ready Due to Supply (NORS). Change to 5.6.12.
5.6.27.1 thru 5.6.27.4	Make this 5.6.12.1 thru 5.6.12.4.
5.6.28	SCNs. Change to 5.6.13.
5.6.28.1 thru 5.6.28.9	Make this 5.6.13.1 thru 5.6.13.9.
5.7	Configuration Status Accounting (CSA). Change to 5.6.14.
5.7.1 thru 5.7.4	Make this 5.6.14.1 thru 5.6.14.7.
5.7.4.1	Make this 5.6.14.8.
5.7.5 thru 5.7.7	Make this 5.6.14.9 thru 5.6.14.11.

- Rationale: Restructuring the contents of MIL-STD-973 as recommended will clearly show a logical process of performance.

2.5.8.20 Recommendation 05-05-20

Change to paragraph 3.4.d.

- Recommendation: Add the following sentence to the end of paragraph 3.4.d:

  * "In the case of software, the Allocated Configuration Identification (ACI) also delineates constraints due to the associated hardware."

2.5.8.21 Recommendation 05-05-21

Change to paragraph 3.5.

- Recommendation: Replace existing text in paragraph 3.5 as follows:

  * "3.5 Baseline. A configuration identification formally designated by the Government at a specific time during a CIs life cycle. Baselines, plus approved changes from those baselines, constitute the approved configuration identification. Baselines may be established as a single event, or in an incremental manner. For configuration management, there are normally four baselines established; Functional, Allocated, Product, and Operational.

2.5.8.22 Recommendation 05-05-22

Change to paragraph 3.6.

- Recommendation: In first sentence change "For CIs, an...." to "For hardware CIs, an....".

2.5.8.23 Recommendation 05-05-23

Change to paragraph 3.6.

- Recommendation: In the sixth line of the part of paragraph 3.6 on page 8, change "...of a significant number of routine software..." to "of a number of software...".

2.5.8.24 Recommendation 05-05-24

Change to paragraph 3.12.

- Recommendation: In the second line of paragraph 3.12, change "...and printouts, which..." to "...and any media, which...".

2.5.8.25 Recommendation 05-05-25

Change to paragraph 3.19.

- Recommendation: Change the subparagraph designations as follows:
  * from "b." to "d."
  * from "c." to "b."
  * from "d." to "c."

2.5.8.26 Recommendation 05-05-26

Change to paragraph 3.41.

- Recommendation: Delete the last sentence of paragraph 3.41.

2.5.8.27 Recommendation 05-05-27

Change to paragraph 3.45.

- Recommendation: In the 1st line of paragraph 3.45, delete the word "initial".

2.5.8.28 Recommendation 05-05-28

Change to paragraph 3.61.

- Recommendation: In the first line of paragraph 3.61, delete the word "current".

2.5.8.29 Recommendation 05-05-29

Change to paragraph 3.72.

- Recommendation: In the second sentence of paragraph 3.72, change "A complete system includes all equipment,..." to "A system may include all equipment,...".

2.5.8.30 Recommendation 05-05-30

Change to Paragraph 4.2.

- Recommendation: Delete subparagraphs a. and c. and interchange paragraphs e. and f.

2.5.8.31 Recommendation 05-05-31

Change to Paragraph 4.4.

- Recommendation: Change "This identification shall consist of..." to "This identification may include, as appropriate,..." and "documents (VDD) and other..." to "documents (VDD), other...".

2.5.8.32 Recommendation 05-05-32

Change to Paragraph 5.3.2.3.1.d.

- Recommendation: Change "...documented in accordance with the requirements of DOD-STD-2167 and as..." to "...documented using DOD-STD-2167 as a guide or as...".

2.5.8.33 Recommendation 05-05-33

Change to Paragraph 5.3.3.1.b.

- Recommendations: In the first line, change "Each discrete CI that is interchangeable shall be..." to "Each discrete hardware CI shall be ...". Delete the second sentence. On page 31, delete "and unique mnemonic" from the first line.

2.5.8.34 Recommendation 05-05-34

Add new Subparagraph 5.3.3.1.g.

- Recommendation: Add the following new subparagraph to paragraph 5.3.3.1:

  * "g. Each software CI shall be identified (alphanumerically) in accordance with the provisions of the contract or in contractor format as appropriate."

2.5.8.35 Recommendation 05-05-35

Change to Paragraph 5.6.19.

- Recommendation: In the second line on page 64, change "...into the PCI." to "...into the CI." In the second sentence on page 64, change "Class II changes are applicable during production and are normally..." to "Class II changes are normally...".

2.5.8.36 Recommendation 05-05-36

Change to Paragraph 6.3.

- Recommendation: Add MIL-STD-1521 to the list of documents superseded by MIL-STD-973.

2.5.8.37 Recommendation 05-05-37

Change to Appendix A.

- Recommendation: Delete paragraph 50.2.

2.5.8.38 Recommendation 05-05-38

Change to Appendix A.

- Recommendation: Change the first line of paragraph 50.8 to:
  * "50.8 NAME: Hardware Configuration Item Identification."

2.5.8.39 Recommendation 05-05-39

Change to Appendix A.

- Recommendation: Delete the last 2 lines of text from subparagraph 50.8.d.

2.5.8.40 Recommendation 05-05-40

Change to Appendix G.

- Recommendation: Delete "drawing" from 40.1.2.1.c.

2.5.8.41 Recommendation 05-05-41

Change to Paragraph 4.4.2.

- Recommendation: In the next to last sentence, change "...system or CI..." to "...system, segment or CI...". Change "...that specification..." in the same sentence to "...the appropriate specification(s)...".

2.5.8.42 Recommendation 05-05-42

Change to Paragraph 4.6.

- Recommendation: Add the following sentence to paragraph 4.6:

  * "The CM discipline is required in the configuration process."

2.5.8.43 Recommendation 05-05-43

Change to Paragraph 4.7.

- Recommendation: In the first sentence of paragraph 4.7, change "Configuration change control measures shall be applied to each approved configuration..." to "The contractor shall apply configuration change control measures to each approved configuration...".

2.5.8.44 Recommendation 05-05-44

Paragraph 4.7.1.

- Recommendation: Recommend the contents of paragraph 4.7.1 be addressed in Section 5 and that the paragraph be deleted from Section 4.

- Rationale: If ECP is covered in paragraph 4.7.1, then deviations, waivers, SCNs, etc., will also need subparagraphs. Deleting paragraph 4.7.1 from Section 4 will keep from segmenting the section.

2.5.8.45 Recommendation 05-05-45

Change to Paragraph 4.8.

- Recommendation: In the second sentence of paragraph 4.8, change "At a minimum,..." to "As a minimum,...".

2.6 PANEL VI FINAL REPORT: SOFTWARE REUSABILITY

2.6.1 INTRODUCTION

Software reusability is a difficult process to implement within the DoD structure, primarily because there are no effective incentives which encourage the creation of the types of assets necessary for reuse. Software reusability concepts to date do not represent a revolutionary technological advance but rather an incremental, methodological, and evolutionary advance to an existing process.

2.6.2 OBJECTIVES

Identify the problems which are perceived to bar widespread cost-effective software reuse in DoD systems.

Provide recommendations on which action can be taken for solving these problems.

2.6.3 BACKGROUND

It has been said that the surest way to avoid software development costs is to avoid developing software--just reuse the existing software. Monterey II (Appendix B) was the previous formal JLC initiative to address these issues. This initiative provided the framework to establish DOD-STD-2167A and to accept the concept of a single high order language as the cornerstone to achieving the reusable asset goal. Monterey II apparently assumed that reusable software assets could be applied across multiple domains within the DoD, but did not stress reusable concepts within specific families of domains. Today, business issues limit DoD's ability to exploit reusable software assets.

Reuse is perceived by many as the use of a software asset in an application other than that for which it was originally developed. The panel believes that this definition is too narrow and that it applies principally to laissez-faire or opportunistic reuse (see Terminology, section 2.6.5). It precludes the concept of expressly developing software assets for use in a number of applications or a family of such systems in a domain.

Attachment VI-B identifies and summarizes many of the other recent software reusability study reports considered by the panel.

2.6.4 SCOPE

The panel focused on identifying the barriers to software reuse and the identification of actions that can be taken to remove them or mitigate their impact. Because of time limitations, the

panel identified what it perceived to be the primary barriers and concentrated on them.

2.6.5 ASSUMPTIONS, CONSTRAINTS, and TERMINOLOGY

This section states the assumptions of the panel's study and of its resultant recommendations. It provides some basic terminology (such as for "software reuse"). Finally, the principal constraints imposed by the study panel on its work are identified.

- Assumptions

  * Software reuse is viewed as a means to the end of reducing software cost, enhancing software quality, and reducing the time to develop a software system. Reuse is not an end in itself.

  * Software reuse is also of great interest because it can help solve the dilemma that increased amount of software needed in defense projects may exceed the nation's capacity to produce the required software.

  * The creation of reusable software assets should be viewed as a capital investment which may be evaluated in terms of ROI. From the government's point of view, capital investment should be viewed with respect to the overall cost reduction achieved for the taxpayer by the amortization of the costs for domain analysis and reusable assets applied over the programs to which they pertain. From a contractor's point of view, ROI should be determined with respect to an investment that he might make in reusable software assets and the profit that he could make in applying them.

  * The Government should not interpret the evolution of systematic software reuse to include premature or revolutionary initiatives, such as absolute (unqualified) mandates relating to reuse methodology.

  * There are both technical and non-technical barriers to reuse. Both are important. It is sometimes difficult to separate the two.

  * "Reuse" includes both the creation of "reusable" software assets as well as their incorporation into an application system.

    * "Reuse" should be viewed as an integral part of the software development process, not as an add-on to it.

- Constraints.

The principal constraints to the conduct of the study reported upon here were:

- The principal objective of the panel was to produce recommendations that could be acted upon by the JLC. Little opportunity was given for the presentation of background material. Because of the complexity of many of the issues considered, this constraint could limit the level of understanding achieved.

- The panel built upon previous work. This helped focus the group's effort by providing points of departure for the discussion.

- The limited amount of time available of necessity limited the deliberations of the panel.

- Terminology

    * Software reuse. The application of a software asset to more than one application system. Reuse may occur within a project (e.g., F-14A, B,....) or across projects (e.g., F-14, ATF,....).

    * Family. A set of systems having a good deal of commonality of function, e.g., a family of aircraft navigation systems.

    * Domain. The functional area covered by a family of systems, e.g., the aircraft navigation system domain.

    * Throughout this report, we refer to reusable software assets as an all-inclusive term, encompassing reusable software architectures, requirements, tools, designs, code, and test plans.

    * There are two principal categories of reuse: systematic and opportunistic (or laissez-faire). Systematic reuse is planned reuse, especially as "planned" for application to more than one system in advance of the creation of those systems. Opportunistic reuse takes advantage of the software assets that happen to be available for the creation of a new system.

* Black-box/White-box. There are two extremes in the ways that a software component (e.g., a unit of code) may be viewed. In the black-box representation the unit is viewed with respect to only its external behavior. In the white-box representation, the internal mechanisms of the unit are visible, thus providing the specific manner in which the unit's behavior is accomplished.

2.6.6 APPROACH

Individual panelists shared reports/presentations from several key on-going reuse activities: National Security Industrial Association (NSIA), Joint Integrated Avionics Working Group (JIAWG), DoD ad hoc Software Reuse Working Group, Software Technology for Adaptable and Reliable Systems (STARS), Strategic Defense Initiative (SDI), Software Development Standard (SDS), and National Aeronautics and Space Agency (NASA). Panelists had been selected in an effort to get a cross-representation of these activities. The panel approach was to build on previous work.

Panelists submitted individual lists identifying high impact barriers to reuse and potential solutions. A modified group technique was then employed to select the highest impact barriers for discussion. Each panelist could cast 15 positive votes and five negative votes with a maximum potential of casting five votes per barrier. Attachment VI-A outlines all the initial barriers and the votes cast per barrier. The group then worked as a whole in a consensus process to discuss highest-priority barriers and identify solutions.

2.6.7 DETAILED REPORT

The report is divided into three subsections. Section 2.6.8.1 identifies recommendations from previous reuse studies that the panel thinks are no longer important or should not be implemented. Section 2.6.8.2 addresses each of the barriers which the panel identified as most significant and provides panel recommendations to removing the barrier as well as the rationale for the recommendations. To the degree possible, the rationales reference the deliberations of other reuse reports that have devoted significant resources to crystallize the arguments. Section 2.6.8.3 lists specific recommendations from this panel.

2.6.8 RECOMMENDATIONS

Panel VI established the following high-level recommendations to address the reuse barriers.

- Government policies and procedures:
  - * Require PEOs to perform domain analysis, establish (through a consensus process between DoD and industry) requirements and software architectures for families of systems and require usage of approved architectures.
  - * Establish a win-win guideline for bi-directional software recovery of investment in reusable software assets (recoupment of government investment and payment of fees to industry).
  - * Use software prototyping to validate architectures, requirements, and the suitability of reusable components.
- Process:

  In order to foster integration of reuse into the overall system/software process,
  - * Promote the view of an application as a member of a family of systems.
  - * Foster integration of domain analysis into the early stages of the acquisition process (prior to where DOD-STD-2167A applies).
  - * Foster process changes to allow generic designs to influence selection of specific application requirements.
  - * Establish guidelines for creating reusable assets and synthesizing systems from assets.
- Technology:
  - * Initiate a technology action plan to define a means for precise software asset description (equivalent to a hardware specification sheet) and over time work to create an industry standards group to evolve this description standard.

- Education:

  * Develop a guidebook and training program for acquisition personnel dealing with nuances of negotiating COTS licenses.

2.6.8.1 REVIEW OF OTHER REUSE STUDIES

The panel endorses many of the summary recommendations of the JLC 1981 Reuse Panel, but concern was expressed that action(s) had not been taken. Specific areas of deviation from the 1981 recommendations are summarized as follows:

- Panel VI does not endorse the establishment of a separate Reusable Software Organization (RUSO) within a company. This is contrary to our recommendations on integrating reuse into the overall process.

- Panel VI does not believe that a lack of library technology is a significant barrier the JLC needs to address. The technology base is evolving; at least seven initial reuse libraries were demonstrated at TRI-Ada '90.

- Panel VI strongly disagrees with the 1981 recommendation to standardize development environments. A standardized development environment is not necessary for systematic reuse to be widespread. What is needed is an emphasis on "plug and socket" interface standards.

The JIAWG was formed to promote (hardware) commonality and (software) reuse within and among three tactical aircraft programs, Advanced Tactical Fighter (ATF) (Air Force), LH (Army), and A-12 (Navy). The JIAWG software task group produced a number of studies including "Contract Elements for Software Reuse" (reference, Attachment VI-B) which provided a proposed contractual/legal structure for incentivizing reuse. This study was considered by the San Antonio I panel. JIAWG also proposed a management and operational approach for a reuse program which would include a "Reuse Office" to act as the office of primary responsibility for managing reuse. The San Antonio I panel believes instead (see Assumptions, section 2.6.5) that reuse should be an integral part of the software process. Responsibility for reuse needs to reside with those responsible for building families of systems in a given application domain. Thus, in DoD, the SAEs could promote reuse in each of the services by working through the PEO responsible for each given weapon system family.

2.6.8.2 BARRIERS AND PROPOSED SOLUTIONS

2.6.8.2.1 Barrier

Reuse is treated as a separate process from software engineering.

- Recommendation: Integrate reuse completely into the lifecycle.

- Rationale: The interest and hype that has been focused on software reuse over the past years has resulted in reuse being viewed as a separate type of development, rather than as an integral factor in every system development. Further, it has also resulted in software reuse being considered as something separate from good system/software engineering principles and practices. Software reuse is a key aspect in good system/software engineering; however, good software engineering practice alone is not sufficient for assuring systematic widespread software reuse. As Government and industry gain experience in software reuse, we will be able to integrate it more effectively into the process, ultimately making it an invisible part of good software engineering. The panel has identified three areas that need the attention of the JLC in order to facilitate the integration of software reuse completely into system software engineering life cycle.

  * Domain Analysis. Implicit in the approach for reaping the benefits of reuse will be viewing a system as a member of a family of systems. Such an approach will provide the basis for reuse, at least, amongst that family of systems. Domain analysis is the process that is used to determine reusable (and potentially reusable) assets that are appropriate to the family of systems. Although it is currently being used by a number of organizations (SEI, U.S. Army Communications and Electronics Command (CECOM), JIAWG, NASA) once the system requirements have been defined, domain analysis also should be used early in the system process. By using domain analysis earlier in the lifecycle, the development and use of high level software assets, such as architectures, requirements, and specifications will be facilitated. The Service PEOs are responsible for families of systems and should also be responsible for domain analysis of families of systems.

  * Prototyping. Prototyping is currently being pursued by a number of agencies as a way of clarifying requirements and minimizing the number

of false starts in a system development. The prototyping concept should be extended to incorporate reuse and to enhance the developers' ability to reuse existing software assets. Reusing assets will enable a user to evaluate the capabilities available through a low cost solution and make trade-offs on additional functionality. Reusing assets can also reduce the time needed in the creation of prototypes, allowing more prototyping to be done and ultimately reducing the development time and cost of systems while improving their overall quality. Thus, prototyping with reusable assets can provide a good vehicle for further clarifying requirements, since stated requirements may be recast or modified, as part of an early ROI trade-off analysis, to allow more reuse of what is available.

* Design for Reuse. The productivity and cost benefits anticipated from reuse can be gained only if the reusable assets are available and the effort required to use those assets does not exceed the savings gained from using them. Designing for reuse is therefore extremely important, for both the design of the assets themselves and the process of creating software from these assets. Guidelines for creating the reusable assets are needed to ensure that the future use of the assets is considered in their design. Guidelines are also needed for creating software using existing assets. When creating high level software assets, it will be important to assure that the necessary components can be developed. It should be noted that some amount of unique software will be necessary to complete the implementation.

Proposed Solutions:

- The JLC should foster integration of domain analysis with the early system development process, requesting the SAEs to issue policy that accomplishes the following:

  * The PEOs should be responsible for domain analysis and creation of families of requirements.

  * The PEOs should review, advise, and be part of the team defining new user requirements (statement of needs) in each PEO's domain to ensure they fit with the evolving family of requirements.

* The representation of a family of requirements (produced by the PEOs' domain analysis) should be used to verify all new requirements posed by the user (i.e., is it a real need or a wish list?).

- The JLC should request the Under Secretary of Defense (Acquisition) to issue a policy to allow contractor participation in early mission activity to do domain analysis for families of systems.

- The JLC should request the SAEs to support service technology base work in the area of domain analysis, particularly in application of domain analysis to the requirements definition phase, and creation of high level reusable assets (such as software architectures) in their Critical Technology Plans.

- The JLC should request the SAEs to have the services sponsor pilot projects to try out existing domain analysis methods (such as NASA's, GSFC's or SEI's) as part of their technology base programs.

- The JLC should request DARPA and the services to initiate a technology action plan to define a means for precise software asset description (equivalent to a hardware specification sheet). The JLC should request NSIA and other industry associations to establish a working group to develop near-term common descriptions for reusable component form and interfaces.

- The JLC should encourage NSIA (and other associations) to conduct the follow-on studies NSIA recommended to identify standards necessary for reuse. The JLC should then request industry standards groups to pursue definition and evolution of these standards.

- The JLC should request the Under Secretary of Defense (Acquisition) to issue policy supporting the use of prototyping in all life-cycle phases to determine which existing assets can be used to implement proposed requirements.

2.6.8.2.2 Barrier

Lack of understood and effective incentives to create, support, and promote use of reusable software assets. (Consolidates Attachment VI-A barriers #4 and #6.)

- Previous studies of how DoD can derive more benefit from software reuse have consistently identified lack of incentives as a barrier to reuse. For example (see Attachment VI-B for references),

* The Software Reuse Strategy Group focused on "inadequate incentives for reuse (both to contractors and program managers)". (SAF/AQXA, P.2)

* The NSIA study said "it is unlikely that DoD mandating software reuse prior to solving many of the non-technical problems would meet with much success. An acquisition approach is needed which encourages both the development and the use of reusable software." (NSIA, p. 6)

* The September 1990 STARS Users' Workshop Nontechnical Reuse discussion group concluded "there is little incentive today for contractors to pursue innovation in the area of reusable software." (STARS/UW, p. 12)

* The draft SDS "Software Reuse Strategy" says "software reuse is difficult because of such non-technical factors as organizational structures, financial disincentives, and lack of specific contractual mechanisms that allow and encourage the creation and use of reusable software." (SDS section 2.2.1)

- After reviewing the studies above, discussions at SA-I reached a consensus that lack of incentives to create reusable software is a bigger problem than lack of incentives to use reusable software assets when they are available. Both Government and contract project managers have well understood incentives to reduce cost, shorten schedules, and improve quality for the project at hand. Given confidence in its quality, performance, and the suitability of its interfaces, project managers will reuse software to help reach these goals.

- On the other hand, a substantial effort is required to create reusable software assets. In addition to normal good software engineering practices, the developers must identify the range of possible contexts and applications where the software will be applied, and make tradeoffs between generality (which expands the domain of applicability) and efficiency in each specific application. This indicates that a substantial investment is required for reuse.

- Two kinds of scenarios were identified where reuse has been widely and successfully employed as an integral part of the software engineering process:

* Multiproject organizations such as Army Tactical Command and Control System (ATCCS) and others listed in (DSD, paragraph 3.1.5) which produce, maintain, and evolve a family of related products, and where there is a technical director/system architect who takes responsibility for seeking out opportunities for commonality and reuse among products of the same generation and between products of successive generations. Examples include Foxboro and IBM MIS (internal).

* COTS tools and COTS components.

- In the first scenario it is important to understand that the purpose of a software architecture "is to provide the structure/interface baseline environment for integrating evolving components" (TRW, p. 4). There are some successful examples of the first scenario of reuse. For example, experience at FCDSSA, Dam Neck (FCDSSA, section 8.1), is based on interface standardization at the architectural level to achieve reuse of components among related systems and between generations of systems.

Proposed Solutions:

Make more use of the scenarios that have been successful in the past. Specifically:

- As noted by previous investigations (e.g., (DSD, p. 64) and (SAF/AQXA, p. 3)), organizations which have continuing responsibility for generations of related systems have been effective at deriving benefits from reuse, so the JLC should request the Under Secretary of Defense (Acquisition) to make reuse incentivization a specific responsibility of PEOs, including:

  * An annual review of reuse plans and results;

  * Involvement of potential contractors in the development of architectures for their domain of responsibility, making approved architectures publicly available, and referencing the architecture in RFPs as appropriate. This is similar to recommendations made by (Kalish), (FCDSSA, para 8.1), and (SAF/AQXA, p. 4); and

  * Use of award fees to promote reuse. Allowing higher award fees for contracts promoting development of reusable software assets and requiring that software assets be reused could be easily structured and would create very little

administrative burden for the Government and industry. Contracts can also be structured to focus on the aspects of reuse most critical to the individual program or class of programs.

- Take steps to encourage the use of COTS for more systems. Specifically:

  * Panel VI recommends, as was done previously by (DSD) and (SAF/AQXA, p. 3), that the JLC should develop a guidebook for contracting officers and project managers which clarifies the nuances of how to license COTS components for use in Mission Critical Computer Systems.

  * The JLC should request the Under Secretary of Defense (Acquisition)(USD(A) to direct the Services to more thoroughly train acquisition personnel in the negotiation of software licenses so as to ensure that only fully qualified personnel represent the Government in such negotiations. This recommendation is supported by results previously reported by (DSD) and (SAF/AQXA, p. 3).

  * The JLC should request the USD(A) to mandate inclusion of COTS component/asset evaluation with experimental prototyping in the requirements definition and architecture definition phases of program plans. See (Kalish) and (SAF/AQXA, p. 3) for previous support of this recommendation.

  * The JLC should request the USD(A) to issue policy that eliminates disincentives discussed in the following section on "recoupment." This recommendation is supported by (Kalish) and (NSIA, p. 18).

The JLC should request the USD(A) to create groups similar to the JIAWG that will serve as a mechanism to facilitate the sharing of reusable software among programs which are in the same domain but report to different PEOs. See (SAF/AQXA, p. 4) for previous support of this recommendation.

The JLC should request DARPA to explore the increased use of domain specific software architecture interface standards as a mechanism which gives companies confidence that assets complying with these interface standards will have a market. This recommendation is supported by (NSIA, p. 9) and (SAF/AQXA, p. 4).

The JLC should request the USD(A) to set up a program to recognize original authors (organization and individuals) of

software assets each time they are actually reused. This should not require any expensive new bookkeeping mechanisms, since DoD CM policy R-9 requires tracking of the authorship of the documents and source lines of code in systems. Various types of recognition employed for non-COTS (i.e., software which is sold only to the Government, so there is no established commercial price) might be award fees, quality awards, and letters of appreciation. Panel VI recommends near-term usage of letters of appreciation which are valued highly by DoD personnel and contractors. Panel VI believes that this concept represents a significant incentive and strongly urges a rapid introduction. For a similar approach, refer to the ideas recommended in (DSD, paragraph 3.1.4).

2.6.8.2.3 Barrier

Recoupment policy inhibits contractors from investing in dual use (government and commercial) reusable software beneficial to DoD.

- Recoupment is a complex issue. NSIA is one of several industry associations attempting to cause a rethinking of recoupment policy. "Industry analysts fear recoupment will substantially threaten the ability of U.S. Defense firms to compete effectively for foreign and commercial business" (W.H. Robinson, President, NSIA, 10 Oct 90). (See also DOD-DIR-2140.2.)

- The application of recoupment to software seems more accident than conscious policy. To our knowledge, no one responsible for recoupment policy has studied the implications of applying recoupment to software, especially as the ranges of possible reuse expand and DoD depends more on COTS and multi-use software components to develop military software. Alternative ways to do so include:

  * Waiving recoupment; or

  * JLC dialogue with NSIA to establish equitable policy that will promote software reuse; or

  * Changing default options so that positive action is required to put recoupment in a contract; or

  * Direct negotiation of recoupment up-front and education of PMs and contracting officers.

- In another recovery of investment issue, when a contractor delivers to the DoD proprietary technology which does not qualify as COTS, the rules for recovering a reasonable return on the contractor's investment are unclear.

Proposed Solutions:

- The JLC should request the USD(A) to promulgate policy mandating up-front negotiation of recoupment in software acquisitions, and in the absence of such negotiations, the contract should not contain a recoupment clause. This policy should be developed in cooperation with industry associations (e.g., NSIA). It should consider recovery of industry investments as well as of DoD investments.

- The JLC should begin a dialogue with NSIA to support the latter's project to establish an equitable recoupment policy and to request NSIA to broaden their investigation to include industry recoupment.

2.6.8.2.4 Barrier

Concerns about confidence and/or lack of quality in reusable assets inhibit reuse.

Rationale:

The central issue is that a developer who wishes to use reusable software must have confidence that the reusable software not only fulfills his requirements, but is of equal or superior quality to the software he can develop. Without this confidence there is every reason to develop the functionality anew rather than to reuse it. This lack of confidence is the underlying cause for the "Not Invented Here" (NIH) syndrome. Additional factors that support confidence in reusable software are (1) the presence of reliable maintenance for the reusable assets; and (2) traceability back to the responsible organization(s).

It must be recognized that the proposed "model" for reuse has a major impact on the above mentioned issues of maintenance and responsibility. The responsibility for maintenance is unclear initially in a GFE-style model, and this causes uncertainty. On the other hand, an intra-company library by comparison has more clear lines of responsibility and avoids many legal issues, which instills greater confidence.

Another relevant aspect of the reuse model is the degree of visibility into reusable assets, i.e., "black-box" or "white-box". White-box reuse is more appropriate for unplanned or opportunistic reuse, since extensive modifications may be needed for the new use. Black-box reuse is designed to avoid such modification, and by restricting user visibility (and hence losing the confidence of personally inspecting code) must compensate with better descriptive information to inspire confidence.

One measure of confidence is the degree to which the software can be shown to be reliable. We can define at least three ways in which software is said to be reliable:

- Dependable (the most common aspect of reliability): the software will do what it is intended to do.

- Safe: the software will not harm human beings while doing its function (while this issue is relevant, this panel does not have the expertise to address it).

- Secure: the software will not deny service or perform incorrectly due to corruption. This is currently being addressed by SDIO's developing a trusted software development methodology.

This panel focused on the issue of dependability.

Proposed Solutions:

The panel believes that in the long term the most effective form of asset reuse will be "planned, black-box" reuse rather than opportunistic reuse with repeated ad hoc modification. Towards this end, the panel offers the following recommendations:

- The JLC should request the USD(A) to direct the development of an approach to capture the lineage, and track record, of reusable assets as a way of promoting confidence (SDI, Attachment B). This approach has at least two aspects: (a) a record of the developer's historical performance; and (b) a record of a reusable asset's historical performance (in terms of use in other applications, test results, user satisfaction, record of errors and fixes). It should be possible to enhance confidence in a reusable asset in this way without requiring a close examination of the process used to develop the asset.

- A relatively standard, concise, accurate, and high-level description is needed to express in a domain-specific way both the interface to an asset and the underlying semantics, side effects, etc. This is essential for instilling confidence in a potential reuser, as it is the only way to reduce the time needed to both find and understand assets, and to successfully match the specification to existing requirements. There are many complex subtleties in the way standard components may be implemented for different operating environments. This fact must be taken into account in an accurate, precise and complete asset description. Further note that this requirement is an element of the infrastructure that is a prerequisite to a central

catalog of reusable software. It is not a call for more asset documentation in greater detail, but rather for more useful, concise information that describes interface semantics uniquely.

In regards to this high-level description there are four specific recommendations to be made:

- The panel is not aware of any analysis having been done on what reusers wish to see included in a complete description of a reusable asset in order to be able to determine if it will be appropriate for the intended application. Such an analysis would be invaluable in creating such a description, and the JLC should direct the SEI to perform it in the near future.

- The JLC should request the SAEs to undertake a near-term effort (based on the above analysis) to establish for one or more initial domains a set of "dimensions" (analogous to those used to describe the components presented in the Grady Booch book "Software Components with Ada") to describe the semantics of the components in those domains. Many of these dimensions may be domain-specific.

- The JPCG-CRM/CSM Subgroup should assess the potential impact of a need for asset descriptions on DOD-STD-2167A DIDs to ensure the DIDs would permit such description.

- The JLC should request industry groups to establish a long-term standards group to evolve the initial asset descriptions over time.

2.6.8.2.5 Barrier

Legal issues regarding ownership and liability present barriers to reusing software assets.

- (SDS, p. 13): "Industry is reluctant to invest in new technology for the Government because of the sweeping data rights demanded by the Government..."

- (NSIA, p. 21): "Data rights issues associated with software reuse are very important and must be resolved."

- The panel felt this barrier worthy of discussion because the panel had split positive/negative opinions about its importance. Everyone agreed the high level of uncertainty surrounding these issues lowers confidence in the feasibility of reuse and increases

risk, inhibiting development and use of reusable software. Those who argued the panel should not address these issues did so not because they believed the issues unimportant but because they were doubtful successful resolution could be achieved.

- The perceived legal problems regarding software reuse and ownership come from several perspectives. The combination of the intellectual property law, the Government's acquisition regulations, and contract law makes the issue complex. This has led to a perception that there is a substantial legal impediment to achieving widespread software reuse. It is not clear that this is in fact the case. Rather than provide a complete legal brief on the subject here, we refer to SEI's Technical Report ESD-TR-86-206 by Pamela Samuelson.

- Is there a difference between those business transactions that anticipate reuse of the asset and those that do not? In the case of non-reuse transactions, a user generally obtains a license for the use of a software asset, under specified conditions, from the owner of the software. The specified conditions can cover a wide range of circumstances, but both parties to the transaction attempt to accomplish their particular needs and at the same time protect their rights.

- In the case of transactions where reuse of the software asset is contemplated, the same conditions will apply. Both parties to the arrangement have the same goals for reuse transactions as they did for non-reuse transactions. The complication that appears to create the problem occurs when one of the parties is the Government. This is an unnecessary complication if the personnel representing the Government are fully versed on options available to them. It is then possible for the Government and its contractor to establish an agreement that accomplishes their particular needs and at the same time protects their rights.

Proposed Solution:

- To the extent legal risks of reusing software are the same as using first-use software, perceptions of greater risk can be mitigated through education. Therefore, the JLC should request that the USD(A) direct the Services to train acquisition personnel in understanding the options available for acquisition of software assets and the situations that best serve the Government.

2.6.8.2.6 Barrier

Lack of centralized catalog of assets.

- The panel does not recommend the establishment of and support for a centralized catalog/library to support reuse at this time.

- The panel considers the establishment and support of a centralized catalog/library to be premature. This finding is based on the panel's analysis of the current state-of-practice of reuse approaches/techniques and a forecasting of near-term changes in this status.

- After careful consideration the panel finds that current attempts at establishing and operating centralized catalogs/libraries (such as Comprehensive Software Management and Information Center (COSMIC), DECUS, SIMTEL-20, Defense Technical Information Center (DTIC), etc.) have met with limited success. The panel's opinion is that the level of success of current attempts to use a centralized catalog/library in support of reuse is not sufficiently high to advocate the concept.

- A major drawback to the successful creation and use of any type of catalog/library of reusable assets is the lack of a firm and comprehensive approach to the basic infrastructure needed for reuse. This infrastructure includes such items as standards/guidelines for the description/characterization of reusable assets, policies on any value-added services to be provided, user operational procedures, etc. Before the establishment of a centralized catalog/library can be recommended, more work needs to be done on developing and coming to closure on basic reuse-related concepts to support it.

  Proposed Solution:

- The panel feels that as the development and utilization of reusable assets becomes better integrated into the day-to-day software engineering methodologies that the development and utilization of catalogs/libraries will naturally arise. In view of this position the panel feels that the establishment of a centralized catalog/library is currently not a central issue and need not be addressed at this time.

2.6.8.3 PRIORITIZED RECOMMENDATIONS

The recommendations in the sections below reflect two prioritized

lists of actions. Paragraph 7.2.1 identifies and prioritizes immediate action recommendations -- items on which the JLC can and should initiate action immediately (within a month). Paragraph 7.2.2 reflects the additional necessary actions that will place DoD on the path to support mid-to-long term institutionalization of reuse as characterized in Figure 1, Paragraph 7.1.6.3. Action on paragraph 7.2.2 recommendations may require additional strategic planning time or sequencing with previous actions.

2.6.8.3.1 TIMEFRAME FOR INSTITUTIONALIZATION OF REUSE

This section deals with identification of various models or strategies of reuse and characterization of what is achievable and desirable in terms of time-phased institutionalization of the model. Institutionalization, in this context, refers to the SEI technology transition model and implies wide acceptance of the model as the common way of doing business. The underlying panel recommendation is that the DoD emphasize the domain-specific, common architecture based model and focus its resources on the business issues, process and technology necessary to institutionalize that model.

Figure VI-A depicts five distinct models or strategies that the DoD could follow. The degree of DoD control implicit in each strategy decreases as one reads down the chart.

Model of Reuse	0-1 yr	2-4 yr	5-10 yr
Institutionalized GFE Components	<----------	Not Major Thrust	---------->
Mandatory use of Software QPL	<----------	Not Major Thrust	---------->
Centralized catalog	<---------- Some false starts	Not Major Thrust	---------->
Common Architecture	Intracompany (Some) Licensable library (Some)	Intracompany Intercompany (Closed) Intra-PEO	Intercompany Intraservice Components industry - Licensable library (COTS) - Support services (NDI) Black Box
Opportunistic	White Box Public Domain Libraries	White Box	White Box

Figure VI-A Time-Phased Institutionalization of Reuse.

The highest degree of control involves Government Furnished Equipment (GFE) component sets. While GFE software will be viable for selected systems, this panel does not support this strategy as a primary mechanism to making reuse widespread, nor is it recommended that the DoD allocate resources in this direction. GFE software can often provide an excuse for overall system slips.

The next strategy addresses creation and use of a mandatory software Qualified Parts List (QPL). Again, there is significant DoD control and components are reused with some degree of selection permitted. However, the panel does not believe software QPL would be a significant enabler in achieving wide-spread resources and such a strategy should not be a major thrust in DoD.

The next strategy involves a centralized catalog or Library of Congress for software. Assets are under control in a single logical place. However, without an emphasis on common interfaces and the basic infrastructure needed for reuse (e.g., standards and guidelines for the description and characterization of reusable assets), a centralized catalog is premature. The panel believes that today a centralized catalog is often incorrectly viewed as a panacea which will solve reuse problems. More work is needed to develop and come to closure on basic reuse-related concepts first.

Next is a model in which there is consensus on a common architecture and interfaces for a specific application domain. In an architecture-based strategy, the natural components for reuse will be identifiable through an analysis of the application domain. In this model, the architecture and standard interfaces are controlled (although subject to a standards evolution process) but the particular implementation of components is not pre-selected. This thus encourages evolution of variants of components optimized for performance, cost, accuracy, etc. This is the strategy the panel recommends the DoD adopt and focus its resources on.

The last model is an opportunistic or laissez-faire strategy. In the opportunistic model, reuse occurs on an ad hoc basis and modification of assets may not be directly controlled. Model of opportunistic reuse occurs today and will continue into the future. However, this is a model that may not serve to facilitate a systematic predictable increase in quality and productivity. The actions the DoD takes should permit opportunistic reuse, but this panel does not recommend allocation of DoD resources in that area.

In opportunistic reuse, software may be viewed as a white box in which the reuser looks inside the component to understand it. In moving towards domain-specific architecture-based reuse, technology needs to focus on enabling a precise asset description of a software component's external behavior (akin to a hardware specification sheet). Then a true black-box software components industry could be feasible.

In terms of time-phased institutionalization of domain-specific architecture-based reuse, the panel foresees that the degree of community consensus on a common architecture in a given domain will broaden over time. In the near-term, there are instances of domain-specific architecture and assets within companies and organizations (e.g., the Navy's FCDSSA. There are instances of licensable libraries (e.g., Booch, EVB, math libraries). If the JLC acts upon the recommendations in this report, then in the mid-term, architecture-based reuse could be achieved within a PEO's area and as the common way of doing business within a company. The panel foresees limited mid-term intercompany

sharing of proprietary architectures/assets in instances of prime/subcontractor relationships and consortiums. In the chart, this is referred to as intercompany (closed). With strong DoD support, the panel envisions broad-based institutionalization of architecture-based reuse within ten years. This can serve as the basis for a software components industry. The industry would consist of licensable COTS libraries for assets which have dual usage (support both DoD and commercial requirements) and self-sustaining, domain-specific libraries of DoD-specific NDI components which operates as a service industry.

In summary, the specific panel recommendations support a short-term strategy as well as a long-term move to practice based on composition of black-box components. If the JLC and the Services follow through on the recommendations in this report and focus their plans and resources on a domain-specific, architecture-based strategy, then reuse can have a significant impact on lowering costs, improving quality and increasing reliability of DoD systems in the mid-to-long term.

2.6.8.3.2 IMMEDIATE ACTION RECOMMENDATIONS

The recommendations given below reflect a prioritized list of the actions that the JLC can take immediately.

2.6.8.3.2.1 Recommendation 05-06-01

- The JLC should foster integration of domain analysis with the early system development process, requesting the SAEs to issue policy that accomplishes the following:

  * The PEOs should be responsible for domain analysis and creation of families of requirements.

  * The PEOs should review, advise, and be part of the team defining new user requirements (statement of needs) in each PEO's domain to ensure they fit with the evolving family of requirements.

  * The representation of a family of requirements (produced by the PEOs' domain analysis) should be used to verify all new requirements posed by the user (i.e., is it a real need or a wish list?).

2.6.8.3.2.2 Recommendation 05-06-02

- The panel is not aware of any analysis having been done on what reusers wish to see included in a complete description of a reusable asset in order to be able to determine if it will be appropriate for the intended application. Such an analysis would be invaluable in

creating such a description, and the JLC should direct the SEI to perform it in the near future.

2.6.8.3.2.3 Recommendation 05-06-03

- The JLC should request the USD(A) to promulgate policy mandating up-front negotiation of recoupment in software acquisitions, and in the absence of such negotiations, the contract should not contain a recoupment clause. This policy should be developed in cooperation with industry associations (e.g., NSIA). It should consider recovery of industry investments as well as of DoD investments.

- The JLC should begin a dialogue with NSIA to support the latter's project to establish an equitable recoupment policy and to request NSIA to broaden their investigation to include industry recoupment.

2.6.8.3.2.4 Recommendation 05-06-04

- The JLC should request the SAEs to support Service technology base work in the area of domain analysis, particularly in application of domain analysis to the requirements definition phase, and creation of high level reusable assets (such as software architectures) in their Critical Technology Plans.

2.6.8.3.2.5 Recommendation 05-06-05

- The JLC should take steps to encourage the use of COTS for more systems. Specifically:

  * Panel VI recommends, as was done previously by (DSD) and (SAF/AQXA, p. 3), that the JLC should develop a guidebook for contracting officers and project managers which clarifies the nuances of how to license COTS components for use in Mission Critical Computer Systems.

  * The JLC should request the USD(A) to direct the services to more thoroughly train acquisition personnel in the negotiation of software licenses so as to ensure that only fully qualified personnel represent the Government in such negotiations. This recommendation is supported by results previously reported by (DSD) and (SAF/AQXA, p. 3).

2.6.8.3.2.6 Recommendation 05-06-06

- The JLC should encourage NSIA (and other associations) to conduct the follow-on studies NSIA recommended to identify standards necessary for reuse. The JLC should then request industry standards groups to pursue definition and evolution of these standards.

2.6.8.4 PRIORITIZED RECOMMENDATIONS TO ACHIEVE TIME-PHASED INSTITUTIONALIZATION OF REUSE

Recommendations 05-06-07 through 05-06-20 reflect a separately prioritized list of recommendations from section 2.6.8.3. Recommendations listed here may, in some cases, reflect a higher overall priority to effecting widespread reuse than some of the immediate actions requested in Section 2.6.8.3. The two Recommendations lists are complementary and when combined they reflect inputs to a near-term tactical strategy and a longer term plan of action.

2.6.8.4.1 Recommendation 05-06-07

- As noted by previous investigations (e.g., (DSD, p. 64) and (SAF/AQXA, p. 3)), organizations which have continuing responsibility for generations of related systems have been effective at deriving benefits from reuse, so the JLC should request the USD(A) to make reuse incentivization a specific responsibility of PEOs, including:

  * An annual review of reuse plans and results;

  * Involvement of potential contractors in the development of architectures for their domain of responsibility, making approved architectures publicly available, and referencing the architecture in RFPs as appropriate. This is similar to recommendations made by (Kalish), (FCDSSA, para 8.1), and (SAF/AQXA, p. 4); and

  * Use of award fees to promote reuse. Allowing higher award fees for contracts promoting development of reusable software assets and requiring that software assets be reused could be easily structured and would create very little administrative burden for the Government and industry. Contracts can also be structured to focus on the aspects of reuse most critical to the individual program or class of programs.

2.6.8.4.2 Recommendation 05-06-08

- To the extent legal risks of reusing software are the same as using first-use software, perceptions of greater risk can be mitigated through education. Therefore, the JLC should request that the USD(A) direct the Services to train acquisition personnel in understanding the options available for acquisition of software assets and the situations that best serve the Government.

2.6.8.4.3 Recommendation 05-06-09

- The JLC should request the SAEs to have the Services sponsor pilot projects to try out existing domain analysis methods (such as NASA/GSFC's or SEI's) as part of their technology base programs.

2.6.8.4.4 Recommendation 05-06-10

- The JLC should request the USD(A) to issue policy supporting the use of prototyping in all life-cycle phases to determine which existing assets can be used to implement proposed requirements.

- The JLC should request the USD(A) to mandate inclusion of COTS component/asset evaluation with experimental prototyping in the requirements definition and architecture definition phases of program plans. See (Kalish) and (SAF/AQXA, p. 3) for previous support of this recommendation.

2.6.8.4.5 Recommendation 05-06-11

- The JLC should request the USD(A) to direct the development of an approach to capture the lineage, and track record, of reusable assets as a way of promoting confidence (SDI, Attachment VI-B). This approach has at least two aspects: (a) a record of the developer's historical performance; and (b) a record of a reusable asset's historical performance (in terms of use in other applications, test results, user satisfaction, record of errors and fixes). It should be possible to enhance confidence in a reusable asset in this way without requiring a close examination of the process used to develop the asset.

2.6.8.4.6 Recommendation 05-06-12

- The JLC should request the SAEs to undertake a near-term effort (based on the above analysis) to establish for one or more initial domains a set of "dimensions" (analogous to those used to describe the components presented in the Grady Booch book "Software Components with Ada") to describe the semantics of the components in those domains. Many of these dimensions may be domain-specific.

2.6.8.4.7 Recommendation 05-06-13

- The JLC should request the USD(A) to set up a program to recognize original authors (organization and individuals) of software assets each time they are actually reused. This should not require any expensive new bookkeeping mechanisms, since DoD configuration management policy R-9 requires tracking of the authorship of the documents and source lines of code in systems. Various types of recognition employed for non-COTS (i.e., software which is sold only to the Government, so there is no established commercial price) might be award fees, quality awards, and letters of appreciation. Panel VI recommends near-term usage of letters of appreciation which are valued highly by DoD personnel and contractors. Panel VI believes that this concept represents a significant incentive and strongly urges a rapid introduction. For a similar approach, refer to the ideas recommended in (DSD, paragraph 3.1.4).

2.6.8.4.8 Recommendation 05-06-14

- The JLC should request DARPA and the services to initiate a technology action plan to define a means for precise software asset description (equivalent to a hardware specification sheet). The JLC should request NSIA and other industry associations to establish a working group to develop near-term common descriptions for reusable component form and interfaces.

2.6.8.4.9 Recommendation 05-06-15

- The JLC should request the USD(A) to issue a policy to allow contractor participation in early mission activity to do domain analysis for families of systems.

2.6.8.4.10 Recommendation 05-06-16

- The JLC should request DARPA to explore the increased use of domain specific software architecture interface standards as a mechanism which gives companies confidence that assets complying with these interface standards will have a market. This recommendation is supported by (NSIA, p. 9) and (SAF/AQXA, p. 4).

2.6.8.4.11 Recommendation 05-06-17

- The JLC should request the Under Secretary of Defense (Acquisition) to create groups similar to the JIAWG that will serve as a mechanism to facilitate the sharing of reusable software among programs which are in the same domain but report to different PEOs. See (SAF/AQXA, p. 4) for previous support of this recommendation.

2.6.8.4.12 Recommendation 05-06-18

- The JPCG-CRM/CSM Subgroup should assess the potential impact of a need for asset descriptions on DOD-STD-2167A DIDs to ensure the DIDs would permit such description.

(INTENTIONAL BLANK)

ATTACHMENT VI-A
BARRIERS

The panel identified 24 barriers to widespread reuse during an initial brainstorming session. A modified group technique was then employed to select the highest impact barriers for discussion. High impact was considered relative to the capabilities of the panel and the JLC to address the barrier. Barriers were characterized as non-technical (business and management issues), process-oriented, and technology-oriented. Each panelist could cast 15 positive votes and 5 negative votes with a maximum potential of casting five votes per barrier. The table below identifies the barriers and the number of positive and negative votes cast per barrier. This is depicted graphically in Figure VI-A. (Note: in the graphic, the number on top of the bar indicates the barrier number in the table below.) Based on the voting, 7 barriers were selected for detailed discussion. The selected barriers are marked with an "*" below. Barriers selected for discussion are detailed in Section 2.6.8.2. A short description of the barriers not discussed by this panel is given in Section A.2.

Section A.1 Summary of Panel Votes to Select Barriers for Discussion (* = selected for detailed panel discussion/resolution)

Non-technical:		
+11/-2	*1.	recoupment
+3/-3	2.	attitude problem
+4/-1	3.	difficult to reuse across government agencies
+10	*4.	no incentives for contractors to take risks (i.e., profits limited)
+4/-1	5.	lack of investment to get reuse over the hump
+14/-1	*6.	no incentive for program manager to spend money for reuse
+7/-6	*7.	liability questions (legal and copyright)
0	8.	Not Invented Here (NIH)
/-15	*9.	lack of centralized catalog of assets
+4/-1	10.	contracting vehicles don't require reuse--need level playing field
+8	11.	confusion in mixing Government Furnished Instructions (GFI), COTS, proprietary, new development
+3	12.	difficult to reuse across projects
Process:		
+2/-2	13.	development vs. maintenance--development process may impose restrictions inhibiting ROI in maintenance

+11	*14.	quality of assets/confidence in assets
+16	15.	lack of uniform application of good software engineering processes
+17	*16.	software reuse is treated as a separate item (another "ility") rather than being integrated into overall process)
+7	17.	lack of historical info to support reuse of assets
+2/-6	18.	view that reuse runs counter to creativity
+7	19.	lack of common architectures in MCCR domains
+9	20.	lack clear guidelines to generate assets and to build systems from reusable assets
+8	21.	need design experimentation building from components (counter to DOD-STD-2167A top-down)

Technology:

+4/-1	22.	extensibility of architectures based on present
+2	23.	focus to date primarily small scale components
+6/-1	24.	lack of interface stability at "mega-component" level

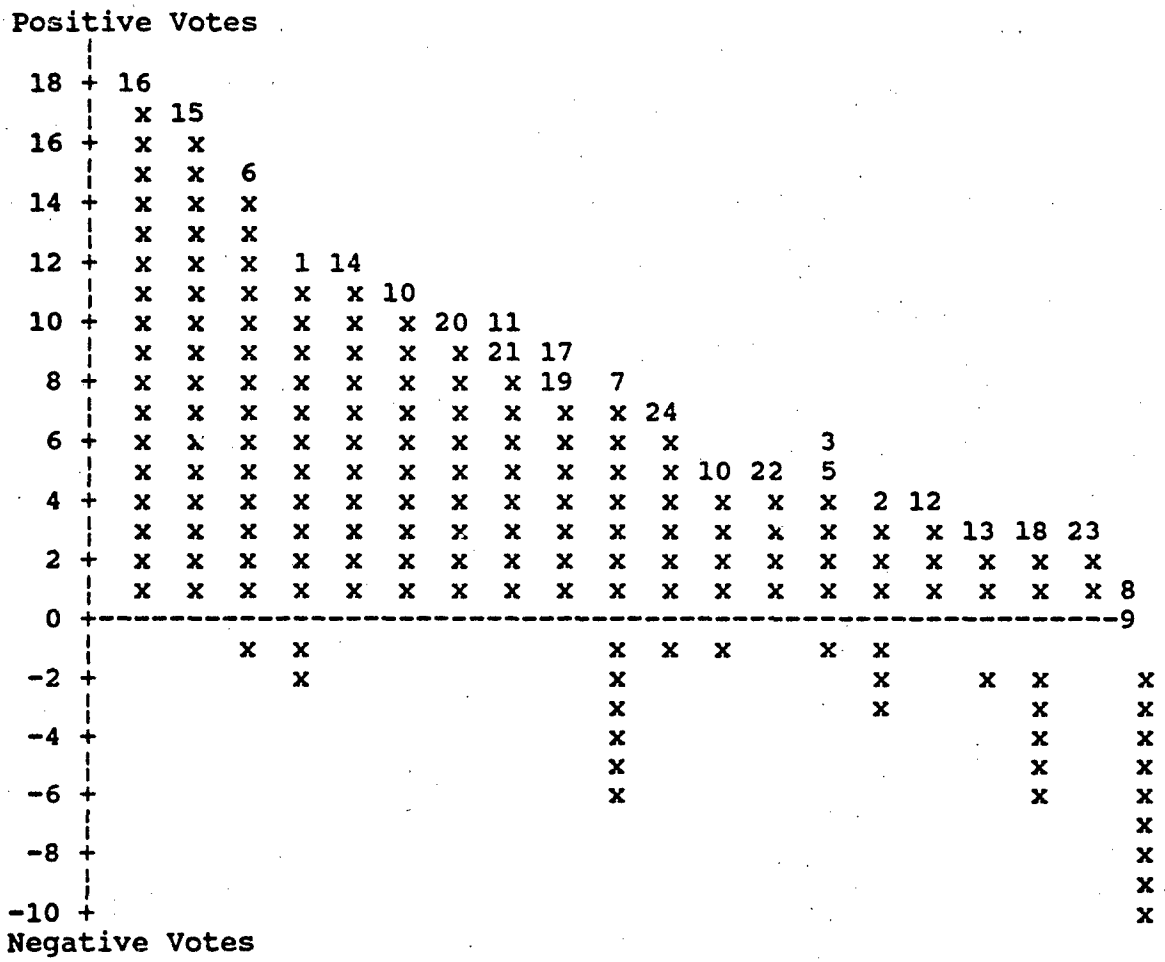

Figure VI-A Panel Votes on Barriers for Discussion

Section A.2 Discussion of Barriers Not Selected for Panel Emphasis.

The barriers identified in A.1 that were not discussed in detail by the panel are briefly described below. Numbers relate back to barrier numbers in section A.1. The panel believed it was better to address some barriers in detail rather than spending time on a broad brush of the issues. The panel encourages others to continue to address and resolve barriers discussed below.

Barrier 2. Attitude problem: A significant barrier to reuse is the possible organizational and culture shock from the imposition of a new technology/methodology.

Barrier 3. Difficult to use across government agencies: There is a significant bureaucratic problem in reusing software developed for one government agency in a new system being developed for another agency. To accomplish this requires specific written permission from the originating agency. A blanket authorization for all government agencies would be one way to remove this barrier.

Barrier 5. Lack of investment to get reuse over the hump: A significant investment in reusable components, domain analyses, and other areas is required before any cost reduction from reuse could be seen. It is difficult to get funds to make this investment to start a reuse program.

Barrier 8. NIH: A primary aspect of "not invented here" (NIH) syndrome is a lack of confidence in software not developed within that organization. This lack of confidence is addressed in section 2.6.8.2.2 of this report.

Barrier 10. Contracting vehicles don't require reuse: Currently, DoD guidance on RFPs and other contracting vehicles does not require that contractors either create reusable software assets in the course of development of new systems or reuse existing software assets in these activities. Consequently, a contractor will be inhibited from either of these reuse activities because of their potential increased incremental costs. All contractors should be required to do so in order to compete on a level playing field.

Barrier 11. Confusion in mixing GFI, COTS, etc.: There can be confusion during fee negotiations on both the Government and contractor sides if the contractor bids a mixture of GFI, COTS, proprietary, and new development. This is partially addressed in section 2.6.8.2.2.

Barrier 12. Difficult to reuse across projects: This can be a difficulty because:

a. If reuse is not specifically planned for, even similar projects may still not have sufficiently similar designs to facilitate reuse, because of lack of common domain engineering.

b. Without reuse in mind, consecutive related projects may not be planned to have compatible schedules to allow reuse.

Barrier 13. Development versus maintenance--requirements for reuse during the maintenance phase must be considered during the original development of reusable assets: The software development tasks performed during the development and maintenance phases of the software life cycle seem to be the same, but there are some differences that can affect software reuse. During the development phase, reused and newly developed assets are integrated to create the new application. During the maintenance phase, this is complicated by the existence of a complete application which must be used. Any reused software not only must perform the functions desired, but also must not require major changes in the existing application. When creating reusable assets, some of the reuse will occur during the maintenance phase, and that reuse forms a part of the return on investment (ROI) anticipated. To ensure the ROI is achieved, the constraints imposed by the maintenance phase must be accounted for in the original development of the reusable asset.

Barrier 17. Lack of historical information to support reuse of assets: This was addressed in section 2.6.8.2.4

Barrier 18. View that reuse runs counter to creativity: Many software engineers do not want to incorporate the work of other engineers into their products. Being required to do so runs counter to their concept of creativity.

Barrier 19. Lack of common architectures in MCCR domains: The existence of common, standard architectures for other-than-management information systems domains would allow reusable components which would meet the interfaces specified by those architectures to be built commercially.

Barrier 20. Lack clear guidelines to generate assets and to build systems from reusable assets: There is no set of generally accepted, clear guidelines available to the software engineering community to provide guidance for either the generation of reusable assets or for their application.

Barrier 21. Need design experimentation building from components (counter to DOD-STD-2167A top-down): There is currently very

little real experience in the DoD community with constructing systems using reusable components, so the approach is relatively untried and as a result has some significant risk attached to it.

Barrier 22. Extensibility of architectures based on present understanding of requirements: Architectures are developed to satisfy sets of requirements and usually have the flexibility to accommodate some change or variation in these requirements. The set of requirements is based on the present understanding of those requirements and the probable change that can be reasonably predicted for the future (usually mid term). When new requirements that fall well outside the domain of the architecture are proposed, the architecture has to be redesigned. To increase the reusable lifetime of the architecture, the architecture must have the capability to accommodate unprecedented new requirements without a major modification.

Barrier 23. Focus to date primarily small scale components: Some of the best work in reusable component design may be found in Common Ada Missile Package (CAMP) or in the "Booch components" (Booch), but these source code components are relatively small pieces of systems. Little work has been done to try to produce large-grained reusable software components.

Barrier 24. Lack of interface stability at "mega-component" level. Little work has been done to define appropriate, much less standard, interfaces for large-grained components.

ATTACHMENT VI-B
REFERENCES

Booch, Grady, "Software Components with Ada." 1987.

(CAMP) Common Ada Missile Packages (CAMP) Studies.

(DSD) DSD Laboratories Draft, Technical Report to STARS/Unisys, "Current Federal Acquisition Regulations and Budget/Finance Environments," 21 December 1990. Final report due on or about 1 March 1991.

(FCDSSA) FCDSSA Dam Neck, "Introduction and Motivation for Software Architecture of Cruiser Upgrade 2000," (date).

(IEEE) IEEE Computer Society, Draft "Standard for Software Productivity Metrics" (P1045/D4.0), 20 December 1990. Provides definitions for "new" and "reused" code and describes current proposals for how to count such statements.

(JIAWG) JIAWG Software Task Group, Draft "Contract Elements for Software Reuse", 13 June 1990. Details a contract strategy to achieve reuse in the Joint Integrated Avionics Working Group (JIAWG) Full Scale Development Request for Proposals.

(JLC/Monterey II) Joint Logistics Commanders Software Workshop, "Final Report (Panel E: Software Reusability)". Monterey, CA: June 22, 1981.

(Kalish) Kalish, Candace, "Using Off-The-Shelf Software in Command, Control, and Communications (C3) Systems", address to Joint Logistics Commanders San Antonio I Workshop, 29 January 1991.

(NASA) NASA Goddard Space Flight Center, "Overview of Software Engineering Methodology Using Reuse Concepts", January 1991. Presents an overview of current prototypes of advanced software engineering environments. These prototypes support such reuse concepts as design rationale capture, system tailoring using reusable designs/specifications, and mechanisms for development of reusable components. (Walt Truszkowski, NASA/Goddard, (301) 286-8821)

(NSIA) NSIA Software Reuse Committee - The NSIA Command, Control, Communications, and Intelligence Committee (C3IC) and the NSIA Software Committee (SWC) formed a joint subcommittee to address business issues associated with DoD acquisition and use of reusable software. Final report was dated 15 December 1990. (Chair: Jack Cooper, Anchor Software)

Parnas, David L., "On the Design and Development of Program Families". IEEE Transactions on Software Engineering SE-2, 1976.

(RADC) Productive Data Systems, "Software Reusability: A Data and Analysis Center for Software (DACS) State-of-the-Art Report". Rome Air Development Center Contract Number F30602-89-C-0082, August 1990.

(SAF/AQXA) SAF/AQXA Report of the DoD ad hoc Software Reuse Strategy Group, Office of the Assistant Secretary of the Air Force (Acquisition), May 24, 1989. Summarizes findings and recommendations of a DoD-wide group of approximately 40 DoD and FFRDC personnel convened in March 1989 to develop a near-term strategy for exploiting software reuse in the DoD.

(SDI) SDI Legal Issues Workshop Report, in publication.

(SEI/FODA) SEI, "Feature-Oriented Domain Analysis," Technical Report, 1990.

(SEI) Samuelson, Pamela, and others, "Legal Issues Report", SEI Technical Report ESD-TR-86-206, August 1986.

(SIGAda) SIGAda Reuse Working Group. The ACM Special Interest Group for Ada (SIGAda) formed a working group (REUSEWG) to examine Ada reuse issues. The group issued a report at the SIGAda Summer meeting on 23 August 1990. Presentations included a report from the Legal Issues Subgroup, an examination of reuse repository capabilities, release of a draft Design for Reuse Handbook, and a report on Reuse Libraries. (Chair: Dennis Ahern, Westinghouse)

(STARS/UW) STARS Reuse Workshops - Although somewhat dated, the reports from early Software Technology for Adaptable, Reliable Systems (STARS) workshops on reuse give a broad perspective on the issues surrounding reuse. The workshops were held in the summers of 1985 and 1986. (Chair: Elizabeth Wald, Naval Research Lab) A more recent STARS Users' Workshop in September 1990 produced a formal report (SE TR CMU/SEI-90-TR-32, ESD-90-TR-232, December 1990) that brings the earlier notes into currency. (Chairs: Dennis Struble, Intermetrics, and Ron Green, U.S. Army Strategic Defense Command).

(SDS) Strategic Defense System, "Software Reuse Strategy," November 1990, and "Software Reuse Action Plan," November 1990. Detailed discussion of management strategies, organizational aspects of reuse policies, contractual and legal issues, software reuse process, and technical strategies.

(TRW) Royce, Walker, "Reliable, Reusable Ada Components for Constructing Large, Distributed Multi-Task Networks: Network Architecture Services (NAS)." TRW Technology Series, TRW Defense Systems Group, Redondo Beach, CA, December 1989.

(INTENTIONAL BLANK)

2.7 PANEL VII FINAL REPORT: ADA BINDINGS

2.7.1 INTRODUCTION

This panel addresses technical issues in the use of Ada as the single language of choice for defense related software. The issues are not primarily in the language itself but rather in the ability of software written in Ada to access services provided by existing software (and hardware) components.

Much of this technology has been or will be standardized by the computing community; for example: data base, screen management and operating system interface standards (e.g., Structural Query Language (SQL), X-Windows, Portable Operating System (POSIX). For those technologies, the Ada community, and in particular, the DoD, must cooperate in processes it cannot control.

Other technologies are entirely an Ada interest. For example, the Ada community must rely on itself to produce Ada math functions and numeric decimal editing features.

2.7.2 OBJECTIVES

Identify current Ada interface activities with high payback for JLC involvement. Propose a process whereby needed interfaces can be integrated into system life-cycle efforts.

2.7.3 BACKGROUND

Perhaps the greatest hindrance to the wide use of Ada as the technical foundation of software engineering is the failure to link Ada to existing and emerging technology. The interfaces, or bindings, between Ada and other components must be standardized to achieve Ada's potential for portability, reliability and maintainability. Production quality implementations must be made available to the software engineering organizations producing today's and tomorrow's systems.

- There are several benefits associated with the use of standards on DoD projects:

  * Reduced development and maintenance costs.
  * Reduced staff training costs.
  * Increased interoperability through standard external data formats and protocols.
  * Increased focus for commercial development efforts.
  * Increased leverage in the commercial marketplace.
  * Increased portability across conforming computer systems.
  * Increased potential for reuse.

The use of standards in DoD acquisition projects is growing. It is common to see lists of standards in RFPs or SOWs, but there has been no concerted effort to date to ensure that the specified standards work together. Acquisitions of similar systems often duplicate the effort required to select and use standards, occasionally coming up with incompatible standards to perform similar functions.

The use of incompatible standards to perform similar functions is obviously an inefficient investment in standards technology. Moreover, multiple standards compromise efforts to build interchangeable software components. A component developed for a system that adopts one standard may be unusable on a system that adopts an alternative standard for the same function.

2.7.4 SCOPE

The panel considered all uses of Ada within the DoD, including Ada for embedded systems, command and control systems, information systems, test systems and software development systems. The panel considered both existing Ada standardization efforts and also the process by which Ada interfaces are selected and integrated for use on DoD projects.

2.7.5 ASSUMPTIONS AND CONSTRAINTS

The panel assumed that the resources to fulfill its recommendations can be found and can be found quickly. Furthermore, the panel assumed that the JLC can coordinate funding with the DoD Consolidated Software Initiative and other Office of the Secretary of Defense (OSD)-level programs. Finally, the panel assumes that appropriate coordination with the Ada 9X project will take place.

2.7.6 APPROACH

The panel gathered people from research, industry and the Government directly involved with the needs of the Ada community. Panel members brought forward individual recommendations based on their experience and knowledge.

2.7.7 DETAILED REPORT

2.7.7.1 THE NOTION OF "STANDARD"

With the growth in the commercial computer industry in the United States and the increased capabilities of commercial hardware and software, the DoD can apply many commercial products to military applications. Emphasis on the use of COTS products and reuse of software developed for other DoD systems are some currently recommended techniques to reduce total system development costs within the DoD.

Standards play an important role in reducing total system development costs. Standards allow an application to obtain services from one of several equivalent COTS products, without becoming dependent on a specific vendor's product. Standards provide a framework of portability that promotes the use of existing software.

In this report, the term "standard" includes formal standards, de-facto standards, commercial products, and stable, widely applicable reusable components. Formal standards are developed by standards-making bodies such as International Standards Organization (ISO), ANSI, IEEE, National Institute of Standards and Technology (NIST) or DoD. De-facto standards are public-domain items typically developed and controlled by consortia such as the X Consortium.

2.7.7.2 STANDARD INTERFACES AND ADA BINDINGS

In general, a standard interface defines a set of services provided to an application. In order to use these interface services from Ada, there must be a definition of the interface expressed in terms of the Ada language. This definition is called an "Ada binding" to the interface.

DoD must participate in the definition of standard interfaces and Ada bindings to those interfaces to ensure that the standard supports DoD needs, particularly Ada's needs. Often industry standards are developed from commercial practice that differs from DoD practice in some way. For instance, the POSIX System service standard was developed from several definitions of services for Unix, expressed in the system programming language for Unix systems, "C". DoD participation in that effort would ensure suitability for DoD systems.

2.7.7.3 CURRENT STATE OF ADA BINDINGS

DoD has emphasized the use of COTS products and standards in acquisitions. This emphasis has increased the demand for implementations of Ada bindings to these products and standards. For many of these products and standards, there are no Ada implementations available.

Developing an Ada binding and implementation requires a substantial amount of work. A project manager may find that it is not cost- or schedule-effective to fund that development from within the project. This often leads the project manager to request a waiver.

To support DoD's Ada policy and to amortize the cost of producing these interfaces, it makes sense to fund the development of Ada bindings to commonly used COTS products and standards. Based on the experience of the panel members, we can nominate several

"high-payoff" interfaces, with the recommendation that the JLC fund the development of these standards, Ada bindings, and implementations.

Attachment VII-A provides information on the status of a variety of standards and Ada interfaces.

2.7.7.4 DEVELOPING ADA BINDINGS

The development of high quality Ada bindings requires substantial resources. The ROI for a well-engineered binding is significant, as both implementing and using the binding becomes much easier and less error-prone.

The best software engineers must be actively involved in developing Ada interfaces. A long-term commitment to the task is critical.

Public review is critical to the success of a binding. But it is costly, particularly in terms of development schedule.

During the review of a binding, a prototype implementation may decrease the effort involved in development of production-quality version. Once review is complete, vendor and third-party software houses can build upon the benefits of the review and prototype. The coordination of all parties' energies leads to a lower-risk Ada interface technology; and therefore the DoD will benefit by investing in the review and prototype process.

2.7.7.5 THE CONCEPT OF "PROFILES"

A profile is a set of coherent standards that provide interfaces, services and supporting formats for interoperability and portability of applications, data and people for a specific applications domain.

For example, a profile for Command, Control and Communication (C3) systems might specify IEEE POSIX as its operating system interface, SQL for its data base management interface, GKS for its graphics interface, and X Windows for its user interface. A profile for Information Systems might include IEEE POSIX, SQL, and X Windows, but not GKS, as it has no need for graphics. The Information Systems profile would also include bindings for decimal arithmetic and indexed files.

It is not enough to specify a list of standards applicable to a system. It is also important to ensure that the standards work together. For instance, many C3 systems specify using both the X Windows system and the GKS. The current X Windows standard and the GKS standard are both silent on how the two standards work together. Therefore, a document specifying both standards must also specify how the two standards should interoperate.

2.7.7.6 DEVELOPING PROFILES

A profile is a set of services tailored to a specific domain. This set should be 'necessary', but need not be 'complete'. There should be a clear need for the specified service within the given domain. Given a set of required services, the next step is to select a standard to fulfill that requirement. The selected standard should meet the overall needs of the applications domain. For instance, an operating system for a embedded real-time domain should support precise timing and synchronization events.

Given the changing state of the standards world, it is possible that some requirements of a given profile are not satisfied by an existing standard. In this case, the profile developer can either acknowledge the unfilled requirement, or can start a standardization activity to fill that requirement. Like the development of Ada bindings, profile development also requires high-quality engineers with experience in the applications domain, Ada, and the component standards.

The next step in developing a profile is to study the interaction of the selected standards. This is the key systems engineering effort in developing a profile. The profile will likely constrain and augment its component standards to support this integration.

Systems integration issues can be resolved once, and subsequent users of the profile will not have to duplicate this effort. Since all applications within a given domain use the same set of services, it will be much easier to move applications from one conforming host to another. More importantly, during maintenance, it will be much easier to replace one implementation of the profile with another conforming implementation at little or no cost or risk to the applications software.

A profile provides a framework for the applications domain. It does not completely specify the application within the domain, but provides basic building blocks that are necessary for the application.

It is worth noting that the Navy has taken this approach with its Next Generation Computer Resources program, developing a set of applications profiles and then selecting existing standards or specifying new interfaces to meet the requirements of the profiles. A similar situation exists within the JIAWG, which is developing standards for avionics systems for the Advanced Tactical Aircraft (ATA), ATF and LHX projects. The Army is also pursuing their end with the Sustaining Base Information System (SBIS) program.

2.7.7.7 MANAGING PROFILES

The JLC should initiate a process leading to the establishment of application domain profiles, to be subsequently applied for JLC software management. This process should lead to the specification of operational profiles, and should involve PEOs, and supported by those involved in the DoD Consolidated Software Initiative (STARS, SEI, AJPO).

The rationale for the definition and use of application domain profiles, and the specific involvement of these DoD agencies is clear: the current state-of-the-practice in DoD Ada interface technology has been fragmented, and a high-level, DoD-wide initiative will counter this fragmentation.

A profile should be application domain oriented, and should support the requirements for Ada application development in the domain. In this way, many DoD programs will be able to leverage the investment made in the definition of applicable profiles.

Leadership for the definition of the profiles must fall on the PEOs. The PEO can readily measure the effectiveness of his profile, since each service now has a General Officer level focal point for specific application domains. In coordination with the DoD Consolidated Software Initiative, the PEOs can then work with JLC and OSD level agencies to come up with a joint profile for all DoD systems within a specific domain. Prior to tasking the PEOs, the JLC must refine the application domain profile task, as a component of an overall Ada interface technology initiative.

2.7.7.8 ADA INSERTION INTO TECHNOLOGY RESEARCH

The JLC should extend DoD policy to require that state-of-the-art computer technology is compatible with Ada. This is particularly important in the earlier, introductory stages of computer technology development. This will lead to reduced risk in mature systems and higher predictability in software management. An important tenet of a DoD Ada interface technology initiative will be a plan to manage the insertion of such technology in longer range programs, such as advanced research.

The current state of practice of demonstrating over-the-horizon technologies without Ada leads to a long lag between demonstration and the ability to apply Ada in those technologies. The goal is to ensure that Ada shall be no more difficult to use than any other language in DoD-sponsored technologies. PE 6.2 technology demonstrations should show the use of technology from an Ada application.

2.7.8 RECOMMENDATIONS

2.7.8.1 Recommendation 05-07-01: Create a DoD-wide Ada Interface Technology Initiative (AITI). (Immediate recommendation.)

- The panel recommends the creation of a DoD-wide AITI. The JLC should immediately create a working group whose task shall be to create a plan for such an initiative and ensure its implementation. The panel recommends that such a working group include representatives of the SEI, DARPA, AJPO and the JLC. The main thrusts of the AITI are given by the remainder of this recommendation.

2.7.8.2 Recommendation 05-07-02: Develop Application Profiles within DoD applications domains. (Mid-term recommendation.)

- The AITI should produce application profiles for DoD applications. The JLC, working through the AITI Working Group, should task the service PEOs to develop these profiles. The PEOs have executive responsibility for the applications built on the profiles. Ownership of the profiles by the PEOs ensures the user buy-in without which the AITI will fail. Funding for this activity should be included as a separate budget item in the DoD Consolidated Software Initiative budget. The PEOs should be provided with technical support from the DoD Consolidated Software Initiative agencies (SEI, STARS, AJPO).

2.7.8.3 Recommendation 05-07-03: Accelerate the development of Ada interfaces and their implementation. (Mid-term recommendation.)

- The AITI will accelerate the development of Ada bindings to interface standards, by funding participation in standards activities. Interface standards requiring such participation will be identified in the profile development effort. Each activity will require approximately eight individuals at one-third time for a period of two years.

- The AITI will accelerate the availability of commercially supported Ada interface implementations by funding DoD efforts to provide public-domain implementations. These implementations provide a basis for development of commercial quality products for these interfaces. Implementations requiring this acceleration will be identified in the profile development effort.

2.7.8.4 Recommendation 05-07-04: Accelerate the development of high-payoff Ada interfaces and their implementation. (Near-term recommendation.)

- Drawing from its experience in the field, the panel believes that it can recommend some interfaces which will occur in many of the application profiles. The panel recommends that Ada involvement in the following standards development efforts be increased immediately (listed in priority order):

  * POSIX System Services
  * POSIX Real-Time Services
  * POSIX Network Services
  * POSIX Security Services
  * Information Systems Services support
  * Numerical Functions, particularly IEEE 754 Floating Point
  * Ada/Atlas Testing Language

Funding for these seven efforts will cost approximately $3.25M per year for two years.

The panel recommends increased effort in the development of implementations of the following Ada interfaces (also listed in priority order):

- POSIX System Services
- POSIX Real-Time Services
- POSIX Network Services, including GOSIP and particularly TCP/IP
- X Windows library and toolkit levels, particularly OSF Motif
- Numerical functions, particularly elementary functions
- SQL and the SQL Ada Module Extension (SAME)
- Information System Services support, particularly decimal arithmetic
- GKS/PHIGS/PEX including compatibility with X Windows
- Ada/Atlas Testing Language

Funding for these eight efforts could approach $5M.

2.7.8.5 Recommendation 05-07-05: Ada Technology Insertion. (Long-term recommendation.)

The AITI should recommend changes to DoD policy that encourage the insertion of Ada technology into research, procurements and common use throughout DoD. Specific recommendations:

- PE 6.2 technology demonstrations should show the use of the technology from an Ada application.

- Procurement of COTS software should require Ada interfaces.

- Ada compilers should be available on all computers procured that may be used as development hosts.

- Suppliers of embedded system hardware interfaces should deliver Ada bindings along with their devices.

- Ada bindings to DoD and military hardware interfaces such as MIL-STD-1553, "Aircraft Internal Time Division Command/Response Multiplex Data Bus", should be developed.

(INTENTIONAL BLANK)

ATTACHMENT VII-A
ADA BINDINGS

A.1 POSIX

1. Name. POSIX (Portable Operating System Interface).

2. Sponsoring Organization. IEEE and ISO.

3. Status of Base Standard. See Chart VII-A.

4. Status of Ada Binding. See Chart VII-A.

5. DoD Applications. Virtually all software applications.

6. Importance to DoD. POSIX defines a collection of system services that will be widely supported by computer vendors and third-party software developers. The fundamental goal of POSIX standards is to support applications program portability.

7. Discussion. POSIX is the name for the IEEE standardization efforts to develop portable application interfaces to operating systems. The effort is coordinated with UniForum, X/Open, NIST and other organizations. The resulting IEEE standards follow processes that permit simultaneous standardization as ISO standards. The IEEE standards typically become Federal Information Processing Standard(s) (FIPS) standards. Each POSIX standard is required to include test assertions and test methods that can be used to test implementation conformance.

The POSIX interfaces are based on existing UNIX interfaces, but it is important to note that many non-UNIX systems, such as DEC's VMS, are planning on POSIX compliance. The effort is broken into different areas that cover the range of interfaces that are used to access operating systems:

1. Basic System Services: includes process management, file handling, terminal communications, and basic protection mechanisms.

2. Real-Time Services: includes real-time scheduling, synchronization, and process control.

3. Security Services: includes discretionary and mandatory security interfaces and definitions.

4. User Command Interface: includes command syntax, user services and utilities.

CHART VII-A Status of Posix Effort

POSIX AREA	STATUS OF STANDARD EFFORT	WIDE-USE INTERFACE	WIDE-USE ADA BINDING	ADA BINDING IMPLEMENTATION
BASIC SYSTEM SERVICES	DONE ISO 9945-1 IEEE 100301 FIPS 151-1	ALL MAJOR VENDORS	IN BALLOT, NOT YET STABLE	ONE TRIAL IMPL DONE TO EARLIER REV
REAL TIME SERVICES	STABLE, IN BALLOT	NO	FIRST CUT DONE	NO WORK YET
SECURITY SERVICES	STABLE, GOING TO BALLOT	NO	NO WORK YET	NO WORK YET
PROGRAM GRAPHICAL INTERFACE	NO WORK YET	SOME	SOME	SOME
NETWORK SERVICES	1 YEAR OF WORK, 1 YEAR FROM BALLOT	NO, BUT TCP/IP,X.500	NO WORK YET	NO WORK YET
MAIL SERVICES	6 MO OF WORK, 1 YEAR FROM BALLOT	NO, BUT X.400	NO WORK YET	NO WORK YET
USER COMMANDS	STABLE, IN FINAL BALLOT	ALL MAJOR VENDORS	NOT APPLICABLE	NOT APPLICABLE
SYSTEM ADMINISTRATION	2 YEARS OF WORK, 6 MO FROM BALLOT	ALL MAJOR VENDORS	NOT APPLICABLE	NOT APPLICABLE

5. User Graphical Interface: includes basic windows, toolkits, and User Interface Management Systems.

6. Network Services: includes trans-network transparent file access, process to process communication, directory name services, and generic data exchange.

7. Mail Services: includes user name resolution and mail delivery services.

8. System Administration: includes system control, integrity monitoring, failure recovery, and activity monitoring.

Chart VII-A shows the status for each of the above POSIX standardization areas. It is notable that Ada binding and implementation efforts have not even begun for many of these areas.

While the POSIX effort covers a wide range of operating system interfaces, there are several notable areas that are frequently considered parts of operating systems, but are not considered to be in the domain of POSIX. Specifically, Data Base Management Systems, graphics other than user interface (e.g., GKS, PHIGS), and knowledge based systems are excluded from POSIX.

POSIX standardization efforts also include standard profiles, which bring together a set of standards and narrow the options under those standards. Currently POSIX has defined profiles for transaction processing, super computing, multi-processing, and real-time. The efforts in these areas only began recently. It is expected that these profile areas each might result in more than one profile, depending on how the application domains shake out.

(INTENTIONAL BLANK)

A.2 SQL

1. Name. Database Management Services:

a. Database Management System Component - Database Language SQL,ANSI X3.135.1-1989 (ISO 9075, FIPS 127); Embedded SQL, ANSI X3.168-1989

b. Data Dictionary/Directory Component - Information Resource Dictionary System (IRDS) ANSI X3.138-1989 (ISO DP 10027.N2642 (1988) IRDS Framework; ISO DP 8800 N2132 (1988) IRDS Services Interfaces; FIPS 156)

c. Distributed Data Component - Remote Database Access (RDA) draft specification, ANSI; (ISO DP 9579, N2971)

2. Sponsoring Organization.

a. Database Management System Component - ISO/International Electro-Technical Committee (IEC) Joint Technical Committee 1 (JTC1)

b. Data Dictionary/Directory Component - ANSI X3H4 Database Group and ISO/IEC JTC1

c. Distributed Data Component - Electronic Countermeasures Activity (ECMA) Technical Committee on Database Standards TC22 and ISO/IEC JTC1

3. Status of Base Standard.

a. These standards, developed in 1982 - 1989 by the ANSI X3H2 Database Committee, specify data access capabilities based on "structured query language" and the relational database model. The X3.135 standard specifies data definitions, data manipulation, access control, and limited integrity constraints. An emerging enhanced specification under active development in both ANSI and ISO will include substantial additional features for schema manipulation, dynamic SQL, exception handling, enhanced integrity constraints, transaction management, and data administration.

b. The ANSI standard (ANSI X3.138-1988) and the FIPS are the same document. ISO is working on an IRDS specification. This specification includes user interfaces only. Data dictionary/directory services consist of utilities and systems necessary to catalog, document, manage, and use metadata (information about data).

c. Distributed Data Component - The ISO/IEC RDA specification is currently undergoing draft balloting in ISO/IEC JTC1. The specification is in two parts: Part 1 - Generic RDA, and Part 2 - SQL Specialization. Final adoption is expected in 1992. Vendor consortia, such as SQL-Access and X/Open, hope to have working prototypes operational in 1991 to demonstrate interoperability among different SQL servers.

4. Status of Ada Binding.

a. Database Management System Component - X3.135 deals with SQL independently of programming languages. X3.168 binds or embeds SQL within the programming languages Ada, COBOL, FORTRAN, Pascal, PL/1, and C. This binding method is not endorsed by the DoD. A module language binding has been jointly developed by the AJPO and SEI. Entitled the SQL Ada Module Extensions (SAME), this specification enjoys DoD approval but is not a formally endorsed standard. SEI is coordinating with ISO to establish the same DL as an international standard. The document is now at Working Group (WG) 9 in committee draft.

b. Data Dictionary/Directory Component - There is no current effort to develop an Ada language binding to the IRDS nor are there commercial implementations of such a binding.

c. Distributed Data Component - There is no current effort to develop an Ada language binding to the RDA specification. All current Ada bindings are proprietary in nature.

5. DoD Applications.

a. Database Management System Component - Definitions, management, and query of structured data stored in a relational database management system.

b. Data Dictionary/Directory Component - Data dictionary/directory services consist of utilities and systems necessary to catalog, document, manage, and use metadata (information about data).

c. Distributed Data Component - Used to establish a remote connection between a database client, acting on behalf of an application program, and a database server, interfacing to a process that controls data transfers to and from a database. The goal is to promote interconnection of database applications among heterogeneous environments, with emphasis on an SQL server interface.

6. Importance to DoD. Database management services include the data dictionary/directory component for accessing and modifying data about data (i. e. metadata), the database management system component for accessing and modifying structured data, and the distributed data component for accessing and modifying data from a remote database. The relational data model has been adopted as the standard data model for use in developing applications. Database management systems provide a robust environment of database services and functions. The system designs of software applications are not limited to stand-alone computer systems. The term system now refers to a terminal (dumb or smart) connected by some means, either direct or through an network, to a processor. SQL provides a standard language for designing data bases and manipulating the data in it across multiple applications. Information about this data must be standardized and the method of accessing this data common if DoD is to achieve interoperability and portability of applications across a large range of systems

(INTENTIONAL BLANK)

A.3 X-Windows

1. Name. X Windows.

2. Sponsoring Organization. MIT X Consortium.

3. Status of Base Standard. Within POSIX Standards Working Group.

4. Status of Ada Binding. No Standard or Stable Ada Interface.

5. Discussion. Current release is X11R4. Graphical window systems providing interfaces to users are currently enjoying a rapid increase in demand and use. This increase is primarily due to the release of the MIT X Consortium X Windows System. Within the DoD, programs are often requiring X Windows and different toolkits and extensions such as "the contractor shall provide X Windows, PHIGS, and the MOTIF toolkit." Often times, this creates despair due to the different levels of X and its compatibility with other graphics applications. In addition, at the higher levels of X Windows such as the toolkit level, there are various competing de-facto standards making the progress quite complex. Each level is described below outlining the status of standardization and implementation at each level. A diagram pictorially describing each level follows.

XLib - Communicates with the X System Server through X Protocol. Manipulates X primitives. Provides the calls back to the XServer. It provides mechanisms for efficiency, such as cacheing, and hides the network protocol from the programmer.

Xt Intrinsics - Sets of functions for basic user interface abstractions. It is layered above the XLib to provide the user easier mechanisms to write X applications.

Toolkits - Sets of basic objects (e.g., menus and buttons) built from a combination of XLib and Xt Intrinsics calls. Different widget sets have individual "look and feel". Sample widget sets are Athena, Hewlett-Packard (HP), OSF/MOTIF, and OLIT.

Virtual Toolkits - Common high level interface between application programs and native window systems. Virtual toolkits allow portability of window applications across a variety of different widget sets (e.g., MOTIF, MacTools) and across different platforms (e.g., SUN, Mac, DEC).

UIMS - User interface management systems (UIMS) provide application developers a means to rapidly prototype user

interfaces. Often times UIMS provides a user an ability to construct menus, dialogue boxes, etc., quickly and to generate code for these applications.

A.4 GOSIP

1. Name. Network Services.

a. Government Open System Interconnection Profile (GOSIP), FIPS 146
- Message Handling Service (MHS), X.400 Series, ISO 10021
- File Transfer, Access, and Management (FTAM), ISO 857, 8613, 10026, 8650, 8652, 8653, 9735, 9594
- Virtual Terminal (VT), ISO draft
- X.500 Directory Services

b. Transparent File Access (TFA) - IEEE P1003.8, Draft 3

c. Distributed Computing Services - OSF/1 Network Computing Services (NCS) Remote Process Communication (RPC) - OSF

2. Sponsoring Organization.

a. GOSIP, Version 1 - ISO and Consultive Committee International Telephone and Telegraph (CCITT)

b. TFA - IEEE

c. Distributed Computing Services - OSF

3. Status of Base Standard.

a. GOSIP - GOSIP is essentially a family of protocols and representations. GOSIP Version 1.0 provides a complete transparent, end-to-end data communications capability based on Open System Interconnector (OSI) transport class 4 (TP 4) and Connectionless Network Protocol (CLNP). Version 1.0 provides electronic mail and file transfer access and management applications. It operates over a variety of local and wide-area network technologies. GOSIP Version 2.0 will add remote logon and office document interchange applications, will provide a new network addressing structure to support dynamic routing, will include provision to operate over International Standard Digital Network (ISDN), and allow an optional connectionless transport service to support transparent file access and other applications. Later versions of GOSIP will include substantial added functionality such as distributed directory services, transaction processing, remote database access, dynamic routing, network management, and network security. Key features may vary with specific combinations of vendor products and users. Layer 7 of the OSI Reference Model generally includes components that are considered Application

Service Interfaces. This interface is used by application programmers to access high-level network services such as file transfer, mail, etc.

b. TFA - TFA includes capabilities for managing files and transmitting data through heterogeneous electronic transmission networks in a manner that is transparent (i.e., does not require knowledge of file location or of certain access requirements) to the user. Many functions of TFA are widely available in existing vendor implementations. But some functions rely on the underlying protocols. The eventual TFA specification should overcome this limitation.

c. Distributed computing services include specifications for remote process communications and distributed realtime support in heterogeneous networks. Distributed access services include functional support for submitting, starting, and stopping processes among processors in a network. No specifications exist that define a complete set of functions necessary to provide remote procedure communications for all types of application platforms (i.e., the language independent representation of remote procedure calls).

4. Status of Ada Bindings. There are currently no on-going efforts to develop an Ada language binding to the high-level network services provided by GOSIP protocols.

5. DoD Applications. GOSIP is applicable to all data communications environments where multi-vendor computing is anticipated. GOSIP is based on OSI standards, the world-wide consensus standards for multi-vendor data communications.

6. Importance to DoD. GOSIP will allow interoperability of systems used within the DoD and its international allies. Network services are the standard services needed to transport data between any two service access points within connected Open System Environment (OSE) compliant systems, and to manage the data communications infrastructure.

A.5 Hardware Interfaces

The need for Ada bindings to specialized hardware interfaces and devices is as important as those for off-the-shelf software products. There exists several standard interfaces in embedded systems, both formal and de-facto, including:

- MIL-STD-1553
- RS-422
- IEEE-488
- VME
- SCSI
- Proteus Digital Channel (PDC).

At this time no known Ada bindings exist for these common interfaces.

There also exists special purpose auxiliary processors used in DoD embedded applications systems. Digital Signal Processors (DSP), Parallel Processors, and Application Specific Integrated Circuits (ASIC) are examples of these. Currently no standard interfaces exists for such devices nor is it expected that the developers of such specialized systems are forced into constraints until the technology is recognized and stabilized.

(INTENTIONAL BLANK)

A.6 PCTE

1. Name. Portable Common Tool Environment (PCTE).

2. Sponsoring Organization. European Computer Manufactuers Association (ECMA).

3. Status of Base Standard. PCTE is a Public Tool Interface, an interface suitable for the support of portable software engineering tools; it is the Kernel interface of the KAPSE concept from the Ada Stoneman document. PCTE provides facilities for the management of typed objects in an object base, of UNIX-like processes and of windows for bit-mapped displays. It thus has affinities with object-oriented database management systems, operating systems and user interface management systems. Currently ECMA does not intend to standardize the PCTE user interface management system. The first version of PCTE, known as PCTE 1.4, was defined in November 1986 in the form of C language functional specifications by project No. 32 PCTE. In addition to interfaces for Object Management and Process Management, PCTE 1.4 contains a UIMS which provides a similar level of facility to X Windows System, but is integrated with the Object Management System. It has not been taken up for the process of standardization by ECMA. The current PCTE 1.5 was produced by the PCTE Interface Management Board of the European Commission.

4. Status of Ada Bindings. In 1987 equivalent Ada interfaces to the PCTE were defined. The standard for complete definitions as Ada language bindings. It is planned to be completed during 1990 and then to be proposed to ISO to become an international standard.

5. Importance to DoD. PCTE is an interface for portable software engineering tools and the Kernel interface of the KAPSE concept from the Ada Stoneman document. DoD has mandated use of Ada in all application development. Ada developers require an interface between tools (programs) to include Ada compilations systems and a native operating system so that work can be accomplished within the constraints of Ada. It is intended to be a foundation for the quality of Ada Programming Software Environment (APSE) proposed by Stoneman and still to be realized. It is also being evaluated as a framework for tools integration under the STARS program.

(INTENTIONAL BLANK)

A.7 Decimal Arithmetic

1. Name. Decimal Arithmetic.

2. Sponsoring Organization. ISO WG9 (proposed).

3. Status of Base Standard. None.

4. Status of Ada Bindings. None complete.

5. DoD Applications. Information Systems, particularly financial applications.

6. Importance to DoD:

a. The absence of a standard Ada mechanism for decimal arithmetic and associated functions (edited output, support for common external representations) is a major impediment to a successful transition to Ada from COBOL, and to interoperability of files/databases between COBOL and Ada.

b. Currently, any organization moving from COBOL to Ada must address the decimal arithmetic problem in an ad hoc fashion. This has resulted in duplication of effort, vendor-specific (thus non-portable) solutions, and sacrifice of run-time efficiency. Moreover, the absence of standard support for decimal arithmetic conveys the impression that Ada is not appropriate for financial and other Information Systems applications.

c. Ada has significant potential benefits over COBOL when it comes to the development and maintenance of large systems, but to fulfill these expectations Ada must more directly support the decimal arithmetic capabilities that are intrinsic to Information Systems.

(INTENTIONAL BLANK)

A.8 GRAPHIC STANDARDS

1. Name. GKS and PHIGS.

2. Sponsor. ANSI and ISO.

3. Status of Base Standard. GKS is complete, but a major revision is in progress. PHIGS is complete.

4. Status of Ada Bindings. Ada/GKS binding provided for the upper level of GKS, but not lower levels. Ada/PHIGS binding has been published by ISO and ANSI.

5. DoD Applications. Graphical displays, cartography, user interfaces.

6. Importance to DoD. Transportability of graphics software and compatibility of software with a variety of input/output devices.

7. Discussion. GKS is an American National Standard (X3.124-1985) and an International Standard (ISO 7942) which defines a set of language - and device - independent types and operations that facilitate the programming of 2-dimensional color graphics applications. In order to be used directly in a program, the GKS entitles must be mapped to a particular programming language. Such a mapping, or binding as it is generally called, has been carried out with FORTRAN, Pascal, C, and Ada. The Ada Language Binding to GKS is also an ANSI Standard (X3.124-3) and ISO Standard (ISO 8651-3).

GKS includes the following concepts:

Graphical output primitives - The basic building blocks under GKS are four main primitives. Polyline draws a sequence of connected line segments. Polymarker marks a sequence of points with a designated symbol. Fill Area displays a closed area (polygon). Text displays a character string, and is supported by such attributes as character height, character spacing, character up vector, etc.

Coordinate Systems - GKS distinguishes between user and device coordinates and supports three different coordinate systems. The World Coordinate System (WC) is a device independent system specified in user coordinate space by the application programmer. The Normalized Device Coordinate System (NDC) is an intermediate system used by GKS that maps the WC into a virtual space on the x and y axis from 0 to 1. The Device Coordinate System (DC) is device dependent, and coordinates are expressed in terms of the device being used.

Segments - These are a group of graphical primitives identified by a unique name. Segmentation allows the collection of output primitives to be manipulated as a unit.

Logical input devices - A logical input device is an abstraction of one or more physical input devices. Each logical input device can be operated in three modes: Request, Sample, and Event.

Workstation - This is a GKS abstraction of graphical devices. The graphical workstation provides the logical interface through which the application program controls the physical devices.

GKS also provides an interface to a system for filing graphical image information for the purpose of long term storage and exchange of graphical image information known as a metafile.

PHIGS supports the storage and manipulation of data in a centralized hierarchical data structure, known as the centralized structure store (CSS). The fundamental entity of data is a structure element and these are grouped together into units called structures. Structures are organized as directed acyclic graphs called structure networks. The creation and manipulation of the data structure is independent of the display.

Graphical output on a workstation is produced by traversing a structure identified for display on the workstation and interpreting the structure elements. The workstation maps between four coordinate systems, namely:

a. WC used to define a uniform coordinate system for all abstract workstations;

b. View Reference Coordinates (VRC), used to define a view;

c. Normalized Projection Coordinates (NPC), used to facilitate assemblies of different views; and

d. DC, one coordinate system per workstation representing its display space.

PHIGS supports two and three dimensional primitives, hidden line, hidden surface removal, and viewing transforms which support parallel and perspective transformations.

The graphical information that is input from a device as a result to operator actions is mapped by PHIGS onto six classes of input, each represented by a data type referred to as a logical input value.

(INTENTIONAL BLANK)

APPENDIX A
LIST OF ACRONYMS

ACCB	Aircraft Configuration Control Board
ACE	Air Combat Environment
ACI	Allocated Configuration Identification
ACM	Association for Computing Machinery
ACO	Administrative Contracting Officer
ADP	Automatic Data Processing
ADS	Auxiliary Display Subsystem
AFCSC	Air Force Cryptologic Support Command
AFR	Air Force Regulation
AFSC	Air Force Systems Command
AFSSI	Air Force System Security Instruction
AFSSM	Air Force System Support Manager
AIS	Automated Information System
AITI	Ada Interface Technology Initiative
AJPO	Ada Joint Program Office
AMC	Army Material Command
AMPE	Automated Message Processing Exchange
ANSI	American National Standards Institute
APSE	Ada Programming Software Environment
AR	Army Regulation
ARTEWG	Ada Runtime Environment Working Group
ASAP	As Soon As Possible
ASIC	Application Specific Integrated Circuits
ASOS	Army Security Operating System
ATA	Advanced Tactical Aircraft
ATCCS	Army Tactical Command and Control System
ATF	Advanced Tactical Fighter
BMS	Battle Management System
C3	Command, Control, and Communications
C3IC	Command, Control, Communications, and Intelligence Committee
CALS	Computer-Aided Automated Logistics Systems
CAMP	Common Ada Missile Packages
CASE	Computer Aided System Engineering
CCITT	International Telegraph and Telephone Consultive Committee
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CECOM	Communications and Electronics Command
CI	Configuration Item
CIMP	Computer Security Implementation and Management Panel
CLNP	Connectionless Network Protocol
CM	Configuration Management
CMAG	Configuration Management Advisory Group
COBOL	Common Business Oriented Language
COCOMO	Constructive Cost Model

COSMIC	Comprehensive Software Management and Information Center
COTR	Contracting Officer's Technical Representative
COTS	Commercial Off-The-Shelf
CR	Computer Resources
CSA	Configuration Status Accounting
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSM	Computer Software Management
CSS	Centralized Structure Store
CSSR	Cost Schedule Status Report
CSU	Computer Software Unit
DAA	Designated Approving Authority
DACS	Department of Defense Data and Analysis Center for Software
DAE	Defense Acquisition Executive
DARPA	Defense Advanced Research Projects Agency
DBDD	Database Design Document
DBMS	Database Management System
DC	Defense Communication
DCS	Defense Communication Systems
DEC	Digital Equipment Company
DECUS	DEC User's Society
DI-MCCR	Data Item-Mission Critical Computer Resources
DIDs	Data Item Descriptions
DL	Designated Language
DLA	Defense Logistics Agency
DM	Data Management
DoD	Department of Defense
DOD-STD	Department of Defense Standard
DODCI	Department of Defense Computer Institute
DODD	Department of Defense Directive
DODI	Department of Defense Instruction
DQSO	Defense Quality Standardization Office
DSARC	Defense Systems Acquisition Review Council
DSMC	Defense Systems Management College
DSP	Digital Signal Processor
DTIC	Defense Technical Information Center
EA	Evolutionary Acquisition
ECMA	Electronic Countermeasures Activity
ECP	Engineering Change Proposal
EE	Electronics Engineering
EIA	Electronics Industry Association
EPL	Evaluated Products List
ESC	Electronics Security Command
ESD	Electronic Systems Division
FAR	Federal Acquisition Regulations
FCA	Functional Configuration Audit
FCDSSA	Fleet Combat Direction System Support Activity
FFRDC	Federally Funded Research and Development Center
FIPS	Federal Information Processing Standard

FODA	Feature-Oriented Domain Analysis
FORTRAN	Formula Translation
FQR	Formal Qualification Review
FQT	Formal Qualification Test
FSA	Functional System Audit
FSD	Full Scale Development
FTAM	File Transfer Access and Management
GCA	Ground Control Approach
GFE	Government Furnished Equipment
GFI	Government Furnished Information
GKS	Graphical Kernel System
GOSIP	Government Open System Interconnection Profile
GOTS	Government Off-The-Shelf
GSFC	Goddard Space Flight Center
HDBK	Handbook
HP	Hewlett-Packard
HQ	Headquarters
HWCI	Hardware Configuration Item
ID	Item Description
IEC	International Electro-Technical Committee
IEEE	Institute of Electrical and Electronic Engineers
IGS	Interaction Graphics System
INFOSEC	Information Security
IRDS	Information Resource Dictionary System
IRS	Interface Requirements Specification
ISDN	International Standard Digital Network
ISO	International Standards Organization
JCG-CE	Joint Commanders Group for Communications and Electronics
JCG-CE-CIMP	Joint Commanders Group for Communications and Electronics - Computer Security Implementation Management Panel
JIAWG	Joint Integrated Avionics Working Group
JLC	Joint Logistics Commanders
JPCG-CRM	Joint Policy Coordinating Group on Computer Resources Management
JTC	Joint Technical Committee
KAIS	Korean Air Force Intelligence System
KAPSE	Kernel Ada Program Support Environment
KSOS	Kernel Secure Operating System
KVM	Kernal Virtual Machine
LH	Light Helicopter
LHX	Light Helicopter Experimental
LOCK	Lines of Code "K" (1K)
MCCR	Mission Critical Computer Resources
MHS	Message Handling Service
MIL	Military
MIL-HDBK	Military Handbook
MIL-STD	Military Standard
MIS	Management Information System
MRB	Manufacturing Review Board

NASA	National Aeronautics and Space Administration
NASA/GSFC	National Aeronautics and Space Administration-Goddard Space Flight Center
NATO	North Atlantic Treaty Organization
NCS	Network Computer System
NCSC	National Computer Security Center
NDC	Normalized Device Coordinate
NDI	Non-Developmental Item
NDS	Non-Developmental Software
NGCR	Next Generation Computer Resources
NIH	"Not Invented Here"
NIST	National Institute of Standards and Technology
NORS	Not Operationally Ready Due to Supply
NPC	Normalized Projection Coordinate
NRL	Naval Research Lab
NSA	National Security Agency
NSDD	National Security Defense Directive
NSIA	National Security Industrial Association
OCD	Operational Concept Document
OS	Operating System
OSD	Office of the Secretary of Defense
OSE	Open Systems Environment
OSF	Operational Support Facility
OSI	Open System Interconnect
PBL	Product Baseline
PCA	Physical Configuration Audit
PCI	Product Configuration Identification
PCTE	Portable Common Tool Environment
PDI	Privately Developed Item
PDR	Preliminary Design Review
PDSS	Post Deployment Software Support
PE	Program Executive
PEO	Program Executive Officer
PEX	Programmers Hierarchical Interactive Graphics Standard Extension for X
PHIGS	Programmers Hierarchical Interactive Graphics Standard
PM	Program/Project Manager
POA&M	Plan of Action and Milestones
POSIX	Portable Operating System Interface for Computer
PROMS	Programmable Read-Only Memories
QPL	Qualified Parts List
R&D	Research & Development
RADC	Rome (NY) Air Development Center
RDA	Remote Database Access
REUSEWG	Reusability Working Group
RFP	Request for Proposal
RFQ	Request for Quote
RI	Revision Issues
ROI	Return on Investment
RPC	Remote Process Communication

RR	Revision Request
RTE	Run Time Environment
RTM	Requirements Traceability Matrix
RTS	Runtime System
RUSO	Reusable Software Organization
SA-I	San Antonio I Software Workshop
SACDIN	Strategic Air Command Defense Information Network
SACLANT	Strategic Air Command (Atlantic)
SAEs	Service Acquisition Executives
SAF	Secretary of the Air Force
SAME	Structured Query Language Ada Module Extension
SBIS	Sustaining Base Information System
SCN	Specification Change Notice
SCP	Systems Concept Paper
SCSI	Small Computer Software Interface
SDD	Software Design Document
SDF	Software Development File
SDI	Strategic Defense Initiative
SDIO	Strategic Defense Initiative Organization
SDP	Software Development Plan
SDR	System Design Review
SDS	Software Development Standard
SECNAVINST	Secretary of the Navy Instruction
SECRETNOFORN	Document Classified Secret, No Foreign Dissemination
SEE	Software Engineering Environment
SEER	System Evaluation and Estimation of Resources
SEI	Software Engineering Institute
SIGAda	Special Interest Group for Ada
SIMTEL-20	Simulated Telephone-20 (Library)
SLIM	Software Life Cycle Model
SLOC	Software Lines of Code
SMTC	Software Measurement Technology Center
SMWG	Software Measurement Working Group
SOW	Statement of Work
SPR	Software Problem Report
SPS	Software Product Specification
SQL	Structured Query Language
SQP	Software Quality Plan
SQPP	Software Quality Program Plan
SRR	System Requirements Review
SRS	Software Requirements Specification
SSA	Software Support Activity
SSDD	System/Segment Design Activity
SSMTC	Service Software Measurement Technology Center
SSR	Software Specification Review
STARS	Software Technology for Adaptable and Reliable Systems
STD	Software Test Description
STE	Software Test Environment
STP	Software Test Plan

SW	Software
SWC	Software Committee
TCB	Trusted Computer Base
TCP	Task Change Proposal
TCS	Trusted Computer System
TCSEC	Trusted Computer System Evaluation Criteria
TEMPEST	Term for EMC/EMI Test Interference Standard
TFA	Transparent File Access
TR	Technical Report
TRI-Ada	Tri-service Ada Committee
TRIGS	Tactical Reconnaissance Information Gathering System
UIMS	User Interface Management System
US	United States
USA	United States Army
USAF	United States Air Force
USD(A)	Under Secretary of Defense (Acquisition)
USMC	United States Marine Corps
USN	United States Navy
UW	User Workshop
VDD	Version Description Document
VME	Versabus Modified for Eurocard
VMS	Virtual Memory System
VRC	View Reference Coordinates
VT	Virtual Terminal
WBS	Work Breakdown Structure
WC	Working Committee
WG	Working Group
WIS	WMMICS Information System
WMMICS	World-wide Military Command and Control System

APPENDIX B
LIST OF PANEL MEMBERS

PANEL 1
SOFTWARE METRICS IMPLEMENTATION

Coordinator:
Maj Dan Romano
HQ AFSC/ENRP
Andrews AFB, MD 20334-5000

Government Co-Chair:
Jack McGarry
NUSC, Code 2233
Building 1171/2
Newport, RI 02841

Industry Co-Chair:
Dick Evans
TRW
MS R2/1150
1 Space Park
Redondo Beach, CA 90278

GOVERNMENT

1. James Rozum
   Software Engineering Institute
   Carnegie Mellon University
   4500 Fifth Avenue
   Pittsburgh, PA 15213-2890

2. Cheryl Jones
   Code 2233
   Naval Underwater Systems Center
   Building 1171/2
   Newport, RI 02841

3. Barry Corson
   NAVAIR, Code 5466
   Headquarters, Naval Air Systems Command
   Jefferson Plaza-2, Room 620
   Washington, DC 20361-5460

4. David Castellano
   ARDEC AMSMC-QAH-A
   Picatinny Arsenal, NJ 07806-5000

5. Anne Quinn
   DCMDM - QTS
   2800 South 20th Street
   Philadelphia, PA 19101-7478

6. Larry Tomenga
   DCMDS - QTD
   1200 Main Street
   Room 510
   Dallas, TX 75202-4399

7. Don Morley
   SA-ALC/LDSA
   Kelly AFB, TX 78241

8. Robert Cochran
   U.S. Army Missile Command
   Attn: AMSMI-QA-QT-SP
   Redstone Arsenal, AL 35898-5290

INDUSTRY

1. Steven Keller
   Dynamics Research Corp.
   60 Frontage Road
   Andover, MA 01810

2. Dr. Robert Charette
   Suite 201
   ITABHI Corp.
   9840 Main Street
   Fairfax, VA 22031

3. Judy Clapp
   MITRE Corp.
   MS A-345
   Burlington Road
   Bedford, MA 01730

4. Ron Willis
   618-B223
   Hughes Aircraft Company
   P.O. Box 3310
   Fullerton, CA 92634

5. David Card
   Director of Software Process and Measurement
   Systems Sciences Division
   Computer Sciences Corporation
   4061 Powder Mill Road
   Calverton, MD 20705

PANEL II
DOD-STD-2167A DIDS: LESSONS LEARNED/ISSUES

Coordinator: Dr. Raghu Singh
SPAWAR, Code 32122
Washington, DC 20363-5100

Government Co-Chair: Gary Weiss
SM-ALC/TIEBB
McClellan Air Force Base
Sacramento, CA 95652

Industry Co-Chair: Fred Yonda
GTE Govt Systems Corporation
77 "A" Street
Needham Heights, MA 02194-2892

GOVERNMENT

1. Bob Calland
   Code 622B
   Naval Ocean Systems Center
   San Diego, CA 92152

2. Raymond Menell
   AMSEL-RD-SE-AST-SE
   CECOM Center for Software Engineering
   Fort Monmouth, NJ 07703-5300

3. Jacquelyn Nixon
   Quality & Control Section
   Ground/Navigation Systems Division
   MCTSSA
   Camp Pendleton, CA 92055-5080

4. Martin Owens
   MS L210A
   MITRE Corp.
   Burlington Road
   Bedford, MA 01730

5. John Hoeferlin
   ASD/RWEX
   Wright-Patterson AFB, OH 45433

6. Wayne Sherer
   HQ ARDEC/SMCAR-BAS
   Bldg 352
   Picatinny Arsenal, NJ 07806-5001

7. Fred Feltman
   DCMDS-HQTX
   805 Walker Street
   Marietta, GA 30060

INDUSTRY

1. Lewis G[illegible]
   Ada Pros, Inc.
   12224 Grassy Hill Court
   Fairfax, [illegible] 22033

2. David Maibor
   David Maibor Associates, Inc.
   P.O. Box 846
   Needham Heights, MA 02194

3. Richard McClellan
   LTV
   208 West Valley View Drive North
   Colleyville, TX 76034

4. Jane Radatz
   Logicon
   P.O. Box 85158
   San Diego, CA 92138

5. Jim Heil
   Dept 74201
   ITT Avionics
   100 Kingsland Road
   Clifton, NJ 07014

6. Marv Lubofsky
   The Aerospace Corp.
   M/S M8-106
   P.O. Box 92957
   Los Angeles, CA 90009-2957

PANEL III
DOD-STD-2168: LESSONS LEARNED/ISSUES

Coordinator:
Don Kosco
HQ AFLC/MMEP
Wright-Patterson AFB, OH 45433-5001

Government Co-Chair:
Chris Neubert
HQ AMC (QA-E)
Room 4S06
5001 Eisenhower Avenue
Alexandria, VA 22333-0001

Industry Co-Chair:
Dick Waina
Hughes Aircraft
Bldg. 807, M.S. G-3
P.O. Box 11337
Tucson, AZ 85734-1337

GOVERNMENT

1. Marshall Potter
   ITAC
   WNY, Bldg. 166
   Washington, DC 20374-5072

2. David Corder
   OC-ALC/LA
   Tinker AFB, OK 73145

3. Gary Odegard
   Software Engineering Institute
   Carnegie-Mellon University
   4500 Fifth Avenue
   Pittsburgh, PA 15213-3890

4. Victor Dias
   DCMAO Attn: GG-QT
   605 Steward Avenue
   Garden City, NY 11530-4761

5. Lorna Ozawa
   HQ DLA
   Alexandria, VA 22304-6100

INDUSTRY

1. Lynn Levine
   Bldg. 618, MS L215
   Hughes Aircraft
   P.O. Box 3310
   Fullerton, CA 92634-3310

2. Jack Cooper
   Anchor Software Management
   Suite 214
   5109 Leesburg Pike
   Falls Church, VA 22041

3. Dick Storch
   Logicon
   4010 Sorrento Valley Blvd.
   San Diego, CA 92121

4. Dusty Rhoades
   Anser
   Suite 800
   1215 Jefferson Davis Hwy.
   Arlington, VA 22202

5. Manny Baker
   Software Engineering Consultants, Inc.
   10219 Briarwood Drive
   Los Angeles, CA 90077

PANEL IV
COMPUTER SECURITY/SOFTWARE INTEGRITY

Coordinator: Ralph Wootton
MCTSSA/AQA
Camp Pendleton, CA 92055-5080

Government Co-Chair: H.O. Lubbes
Naval Research Lab, Code 5540B
4555 Overlook Avenue, S.W.
Washington, DC 20375-5000

Industry Co-Chair: Dr. Fred Maymir-Ducharme
DS-12
Grumman Data Systems
1000 Woodbury Road
Woodbury, NY 11797

GOVERNMENT

1. Capt Bob Pierce
   Air Force Cryptologic Support Command
   Kelly AFB, TX 78241-5000

2. John Preusse
   Center for Command, Control, Communications Systems
   USA CECOM
   Fort Monmouth, NJ 07703-5300

3. Charles Payne
   Naval Research Lab
   Code 5540
   4555 Overlook Avenue, S.W.
   Washington, DC 20375-5000

4. Mario Tinto
   NSA/C-8
   9800 Savage Road
   Fort Meade, MD 20755-6000

5. Capt. W. C. Henderson
   Information Systems Branch Head
   MCTSSA, SD-08
   Camp Pendleton, CA 92055-5080

INDUSTRY

1. Dr. Ronald Gove
   INFOSEC Systems Engineering
   Booz Allen & Hamilton, Inc.
   4330 East West Highway
   Bethesda, MD 20814

2. Lester Fraim
   MITRE Corp.
   7525 Colshire Drive
   McLean, VA 22102-3481

3. Terry Vickers-Benzel
   Trusted Information Systems
   Suite 265
   11340 West Olympic Blvd.
   Los Angeles, CA 90064

4. Richard Powers
   MS 8503
   Texas Instruments
   P.O. Box 869305
   Plano, TX 75086

5. Bonnie Danner
   TRW Systems Integration Group
   One Federal Systems Park Drive
   Fairfax, VA 22033-4417

6. Karl E. Ipsen
   CECOM Center for Software Engineering
   AMSEL-RD-SE-CRM
   Fort Monmouth, NJ 07703

PANEL V
SOFTWARE CONFIGURATION MANAGEMENT

Coordinator: Dana Dovenbarger
OO-ALC/TISAC
Hill AFB
Clearfield, UT 84056-5609

Government Co-Chair: John Holcomb
OC-ALC/MASF
Tinker AFB, OK 73145-5990

Industry Co-Chair: Ron Berlack
Lockheed Sanders
NCA1-6244
95 Canal Street
P.O. Box 2044
Nashua, NH 03061-2044

GOVERNMENT

1. Linda Burgher
   DQSO Technical Data Division
   II Skyline Place, Suite 1401
   5203 Leesburg Pike
   Falls Church, VA 22041-3466

2. Richard Seyfert
   CECOM AMSEL-RD-SE-CRM
   Fort Monmouth, NJ 07703-5300

3. Maj Larry Griggs
   MCRDAC C2AA
   Washington, DC 20380-0001

4. William Henson
   OC-ALC/LASFB
   Tinker AFB, OK 73145-5990

5. Chuck Kuniyoshi
   Code 4081
   Pacific Missile Test Center
   Point Mugu, CA 93042-5000

6. Bruce Dubbs
   SA-ALC/TIFCE
   Kelly AFB, TX 78241-5000

INDUSTRY

1. Ron Pruiett
   COMPTEK Research, Inc.
   4849 Viewridge Avenue
   San Diego, CA 92123-1667

2. Dennis Nickle
   E-Systems, Melpar Div.
   7700 Arlington Blvd.
   Falls Church, VA 22046

3. Johnnye Burnhan
   Compass Corp.
   1040 Broad Street
   Shrewsbury, NJ 07702

4. Jerry Raveling
   System Technology Institute
   115 Eagle Drive
   Killdevil Hills, NC 27948

5. Brett Pope
   12004 Canter Lane
   Reston, VA 22091-2113

PANEL VI
SOFTWARE REUSABILITY

Coordinator(s): Doug Gerritsen
HQ ARDEC/SMCAR-BAS
Picatinny Arsenal, NJ 07806-5001

Capt Marc Hoffman
HQ AFSC/ENRP
Andrews AFB, MD 20334-5000

Government Co-Chair: Jim Hess
HQ DA SARD-ZR
Room 3E432
Pentagon
Washington, DC 20310-0103

Industry Co-Chair: Teri Payton
Unisys
12010 Sunrise Valley Drive
Reston, VA 22091

GOVERNMENT

1. LtCol Rick Gross
   HQ USAF, SAF/AQK
   Washington, DC 20330-5100

2. Ed Gallagher
   Center for Software Engineering
   USA CECOM
   ATTN: AMSEL-RD-SE-AT
   Fort Monmouth, NJ 07703

3. Walt Truszkowski
   NASA, Code 522.3
   Goddard Space Flight Center
   Greenbelt, MD 20771

4. Thomas J. Coneeney
   Code 24
   Fleet Combat Direction Systems Support Activity
   Dam Neck
   Virginia Beach, VA 23461-5300

INDUSTRY

1. Dr. Whit Ludington
   ARINC Research Corp.
   2551 Riva Road
   Annapolis, MD 21401

2. John Gaffney
   Software Productivity Consortium
   2214 Rock Hill Road
   Herndon, VA 22070

3. William Novak
   GE Aerospace
   c/o Software Engineering Institute
   Carnegie-Mellon University
   4500 Fifth Avenue
   Pittsburgh, PA 15213

4. William Farrell
   VP, Government Systems Division
   DSD Laboratories
   75 Union Avenue
   Sudbury, MA 01776

5. William Carlson
   Intermetrics
   733 Concord Avenue
   Concord, MA 02138

PANEL VII
ADA SECONDARY STANDARDS

Coordinator: Gary Petersen
OO-ALC/MMEM
Hill AFB, UT 84056-5609

Government Co-Chair: Maj Tom Croak
HQ USAF/SCXS
Pentagon, Room 5C1067
Washington, DC 20330-5190

Industry Co-Chair: Dr. Marc Graham
Software Engineering Institute
Carnegie-Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213

GOVERNMENT

1. Maj Terry Fong, USA
   ISEC/CEC ASQB-SEP-C
   Ft Huachuca, AZ 85613

2. David Emery
   MS A155
   MITRE Corp.
   Burlington Road
   Bedford, MA 01730

3. Dr. Jack Kramer
   DARPA/ISTO
   Suite 317
   1500 Wilson Blvd
   Arlington, VA 22209

4. 1Lt Cathy Lin
   ESD/AVS1
   Building 1074
   Hanscom AFB, MA 01731-5000

5. Mike Kiernan
   Code 1032
   Naval Air Development Center
   Warminister, PA 18974

6. Capt Kim Langdon
   AJPO
   Room 3E114
   Pentagon
   Washington, DC 20301-3080

7. Russ McKissick
   MCTSSA
   Camp Pendleton, CA 92055-5080

INDUSTRY

1. Ben Brosgol
   ALSYS, Inc.
   67 South Bedford Street
   Burlington, MA 01803-5152

2. Stowe Boyd
   Meridian Software Systems, Inc.
   10 Pasteur Street
   Irvine, CA 92718

3. Steve Deller
   Verdix Corp.
   14130-A Sullyfield Circle
   Chantilly, VA 22021

4. Paul Baker
   CTA, Inc.
   616 Executive Blvd.
   Rockville, MD 20852